

THE NORTON UTILITIES™

*The Award-Winning Standard for
Data Recovery and Protection*

**Disk
Explorer** ♦
♦

Disk Explorer

Limits of Liability and Disclaimer of Warranty

See disk envelope for Limits of Liability and Disclaimer of Warranty provisions.

Copyright Notice

Copyright © 1988—1991 Symantec Corporation. All rights reserved.

No parts of the publication may be copied or distributed, transmitted, transcribed, stored in a retrieval system, translated into any human or computer language, or disclosed to third parties without the express written permission of Symantec Corporation, 10201 Torre Avenue, Cupertino, CA 95014 U.S.A.

Trademarks

Symantec, Calibrate, Diskreet, Speed Disk, and WipeInfo are trademarks of Symantec Corporation.

Norton Disk Doctor, Norton Utilities, UnErase, WipeDisk, and WipeFile are registered trademarks of Symantec Corporation.

Other brands and products are trademarks of their respective holder/s.

10 9 8 7 6 5 4 3 2

Contents

Introduction	I-vii
--------------------	-------

Chapter 1 The Norton Disk Companion: A Guide to Understanding Your Disks

Introduction	1-3
The Hardware	1-3
Tracks	1-4
Double-Sided Disks	1-5
Cylinders	1-5
Seek Time	1-9
Track-to-Track Seek Time	1-10
Access Time	1-10
Average Access Time	1-10
Average Latency	1-11
Transfer Rate	1-11
Sector Addressing	1-11
Physical Formatting	1-13
Must You Interleave Me?	1-13
The Software	1-15
Getting Logical	1-15
The Boot Record	1-16
Clusters	1-18
The File Allocation Table	1-19
Cyclic Redundancy Check	1-24
The Root Directory	1-27
Tying Up the Loose Ends	1-31
Things That Go Bump in the Disk	1-35
Conclusion	1-36

Chapter 2 Disk Editor

What It Does	2-3
When to Use It	2-4
How to Use It	2-5
Before You Start	2-5
Running the Program	2-5
Getting Help	2-6
Configuration	2-6
Selecting an Object	2-8
Selecting a File	2-9
Identifying the Disk	2-10
Selecting a Directory	2-11
Selecting a System Area	2-11
Selecting Clusters	2-12
Selecting Logical Sectors as Objects	2-13
Selecting Physical Sectors as Objects	2-14
Linking to Related Objects	2-15
Print As... ..	2-17
Memory Dump	2-18
Displaying Information	2-19
Object Information	2-19
Drive Information	2-20
Map of Object	2-20
Editing Objects	2-21
Selecting a View	2-21
Making Changes	2-23
Undoing Changes	2-24
Saving Changes	2-25
Creating a New Object	2-25
The Hex Viewer	2-26
The Directory Viewer	2-29
The Boot Record Viewer	2-31
The FAT Viewer	2-33

The Partition Table Viewer	2-34
Power Editing Features	2-37
Tools	2-43
Calling DOS Temporarily	2-44
Terminating Disk Editor	2-45
Menu Reference	2-46
Shortcut Keys	2-46
Object Menu	2-46
Edit Menu	2-47
Link Menu	2-48
View Menu	2-49
Info Menu	2-49
Tools Menu	2-50
Quit Menu	2-51
Operational Notes	2-52
Command Line Use	2-52

Chapter 3 Troubleshooter

Introduction	3-1
Data Recovery Procedures	
Overwritten File Fix	3-3
Recovering Cross-Linked Files	3-6
Lifting Data off a Bad Disk	3-7
Recover from DOS's RECOVER	3-14
Fixing a Trashed Root Directory	3-14
Directory Fixes	
Removing Stubborn Files	3-17
Fixing Bad Directory Entries	3-19
Directory Tree Structure Fix	3-21
Recovering Subdirectories from a Bad Directory	3-25
Removing Stubborn Directories	3-28

System Area Fixes

Boot Record Editing 3-29

Media Descriptor Byte Editing 3-31

Fixing the DOS System Files 3-32

Absolute Sector Level Fixes

Partition Table Editing 3-37

Fixing a Bad Track 0 3-39

Low-Level Format for an XT 3-43

Tables

General Disk Information 3-44

File Attributes 3-45

Boot Record 3-46

Disk Types for IBMs 3-47

Disk Types for Compaqs 3-48

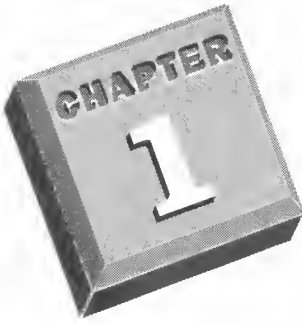
Introduction

The Norton Disk Editor (Disk Editor for short) is an exceptionally powerful tool that lets you access and edit the most sensitive areas of a disk: files, FATs, and directories that control how data is stored on and retrieved from the disk. Like all power tools, it must be used wisely and intelligently. We recommend that you do not attempt to edit any area of any disk—but especially a hard disk—unless you understand what you are doing. You could make all the data on the disk inaccessible to DOS or any other program (except the Disk Editor).

The three chapters in this manual all deal with information you need in order to use Disk Editor wisely and effectively. The second chapter, “Disk Editor,” explains how to use Disk Editor; you should read it in its entirety before trying to edit a disk for the first time. But it does not explain how disks work. If terms such as cluster, FAT, and partition table are new to you, then you should read the “Disk Companion” chapter in this manual first. “Disk Companion” explains concepts which are essential to understanding how Disk Editor works.

When Disk Editor is configured in read-only mode (this is the default setting), you can safely explore disks without changing them. Thus, once you have learned to get around in Disk Editor, you can use it while reading the “Disk Companion” chapter to explore your hard disk and actually see what the text is describing. In the process, you should become more comfortable with the Disk Editor.

The last chapter in this manual, “Troubleshooter,” contains step-by-step instructions for the most common data and disk recovery procedures using Disk Editor. For most disk problems, the Norton Disk Doctor will provide an automated solution. We know there are some of you, though, who prefer to have complete manual control over the process of data recovery and disk repair. The “Troubleshooter” is for you.



The Norton Disk Companion: A Guide to Understanding Your Disks

Contents

Introduction	1-3
The Hardware	1-3
Tracks	1-4
Double-Sided Disks	1-5
Cylinders	1-5
Seek Time	1-9
Track-to-Track Seek Time	1-10
Access Time	1-10
Average Access Time	1-10
Average Latency	1-11
Transfer Rate	1-11
Sector Addressing	1-11
Physical Formatting	1-13
Must You Interleave Me?	1-13
The Software	1-15
Getting Logical	1-15
The Boot Record	1-16
Clusters	1-18
The File Allocation Table	1-19

Cyclic Redundancy Check	1-24
The Root Directory	1-27
Tying Up the Loose Ends	1-31
Things That Go Bump in the Disk	1-35
Conclusion	1-36

INTRODUCTION

This chapter is a brief tour through the inner workings of disks. We'll begin by examining disks and disk drives from a hardware point of view: how they're made, how they work, what the various parts are called. We'll look at the basic "building blocks," such as platters, heads, tracks, cylinders—what they are, how they work, and how they're put together to make a disk drive.

In the second part of the Disk Companion, we'll concentrate on the software that makes disks work. We'll learn about the difference between physical and logical formatting; look closely at clusters, the basic building blocks of files; examine the infamous File Allocation Table (FAT); check out a number of other software factors that affect disk performance; and conclude with a discussion of disk directories and some of the concepts behind UnErasing.

The software section, while occasionally referring back to ideas discussed in the hardware section, is more or less independent. If you're impatient, therefore, you may want to scan the first section and then go directly to the section on software.

Disks are one of the most interesting parts of a personal computer. It's a big subject, so we've tried to pick out just those parts of it that you'll find most useful in your day-to-day work. There are several other related topics—disk management, for example—that are interesting and worthy of examination also, but including them would have diluted the sharp focus of the Disk Companion. To paraphrase the famous tag line from Dragnet: "These are the facts, and just the facts."

THE HARDWARE

Considering the reputation for complexity that disks—particularly hard disks—seem to have, it's surprising how simple the basic concepts behind them really are.

Although you've probably heard it said a score of times, it bears repeating that disks—both floppy and hard—depend on the same phenomenon as audio or video tape recorders to store their data: A recording head magnetizes microscopic particles embedded in a surface; moving the particles past the magnetized head causes the particles to become magnetized.

In an audio tape—and a digital computer tape too, for that matter—the magnetic medium is simply a long string of plastic tape embedded with metal particles. (Incidentally, the most popular metal particle among tape makers is iron oxide, otherwise known as rust. Floppy disk surfaces are brown because they're covered with rust.) But there's nothing magic about that long thin shape. You could successfully create a magnetic recorder with a medium of any shape, provided you could master the mechanics of moving the medium past a recording head.

Tracks

Imagine touching your index finger to an ink pad and then holding it just above a record spinning on a turntable. If you were to touch the spinning record lightly with your inky finger, you'd leave a finger-width ring of ink on the record. Now imagine that the record is a floppy disk and your fingertip is a magnetic read-write head. The inky trail your finger left on the disk would then have an official name: It would be called a track.

By applying a digital signal to the read-write (recording) head, we can record information on the floppy disk track exactly the same way we could record it on a stringy audio tape. The only difference is that, with an audio tape, we would record a continuously variable signal representing a sound waveform; on a floppy disk we'd record either a maximum signal or nothing at all. In other words, on an audio tape we'd record an analog signal, while on a disk we'd record a digital signal.

Back on the circular track on our floppy disk, the physics of magnetic heads dictates that the tracks be fairly thin. That means our single track of data on the disk occupies only a fraction of the total disk area. But just as you can record two, four, or more tracks on a single audio tape, so is it possible to record multiple tracks on a single disk. And just as with audio tapes, we can record multiple tracks in either of two ways: by adding more heads, each positioned to record its track inside (with a smaller radius than) the previous one, or by moving a single head back and forth across the disk.

The first method is expensive but fast, since it allows you to record multiple tracks simultaneously. The second method—stepping a single head across the disk—is slower but far more economical, and it's the method most manufacturers have opted for. (But not all: Some high-performance drives do use multiple heads per disk surface as a way to achieve speed.)

Double-Sided Disks

So far, we've considered only one surface of a floppy disk. But just like records, floppy disks have both a top side and a bottom side, and there's nothing to prevent disk manufacturers from treating both to a magnetic coating, allowing us to record on both the top and bottom surfaces. (In that respect, disks are a bit different from tape, which is coated and recordable on only one side. The opposite is actually true of floppy disks: A disk coated with magnetic material on only one side tends to warp.)

Recording on both sides of a disk yields economies of scale. A disk drive that records on a single side needs a motor to spin the disk, an apparatus to hold the head against the disk, a second motor to move the head back and forth across the disk, a chassis to hold the whole works, and so on. To record on the flip side doesn't require the addition of yet another chassis, motor, or much of anything else. All it requires is a tweak to the apparatus that holds the head—it now becomes a pincer-like affair—and the addition of a second head to the bottom fork of the pincer.

Cylinders

There's another benefit to recording on both sides of a disk: Twice as much data can be written before the head has to be moved from one track to the next. Data can be written first to the track on the top side of the disk; then, without movement of the head, more data can be written to the track on the bottom side. The pair of tracks that lie over each other and can be written without movement of the head assembly are collectively referred to as a cylinder.

For convenient reference, both cylinders and tracks are numbered. The outermost track is called track 0; the track on the top side of the disk is called track 0, side 0, and the track on the bottom is called track 0, side 1. Or you can refer to both track 0s together as cylinder 0. Standard 360K PC floppy

disks have 40 cylinders, numbered from cylinder 0 (nearest the rim) to cylinder 39 (nearest the spindle). (When working with disks, we'll find that numbering generally starts at 0, rather than 1. An exception to this is sectors, whose numbering starts at 1. Clusters are another exception, and a curious one at that: Their numbering starts at 2. We'll discuss clusters shortly.)

Let's finish our tour inside a disk drive. So far, we've mostly been describing the double-sided, two-headed floppy disk drive in fine detail. On a floppy disk, the recording head actually sits directly on the disk's magnetic coating. While this allows the head to read and write the strongest signal possible (the strength of the signal falls off quickly as the distance between the head and the magnetic surface increases), it has some drawbacks. Friction, for example, limits the speed at which the disk can spin. Then there's the vexing problem of keeping the head in constant contact with the flexible disk surface, and the equally vexing problem of designing a head that doesn't try to imitate a chisel. Since the speed of rotation directly affects the reading speed—you can't read bits off the disk any faster than they rotate by the head—floppy disks are thus slow by their very nature.

An alternative approach—the one adopted by hard disks—would be to hold the head slightly above the disk, so that friction between the head and disk, minor irregularities in the disk surface, and wearing of the disk surface by the head all cease to be problems. A whole other set of problems arises with this approach, however. To start with, since the strength of the signal read or recorded to the disk decreases exponentially as the distance between head and disk increases, it's imperative that we keep the head as close to the disk as possible, yet keep it far enough away so that we're in no danger of touching it—either as components heat up and expand or because of imperfections in the disk surface.

The hard disk designer's rather elegant way of meeting these requirements has been to turn the recording head into a miniature airplane—or, more accurately, a miniature glider.

The hallmark of a hard disk system is the layer of air whipped up by the rapidly spinning disk that the head actually floats on. As you can imagine, that layer of air is very thin. At about 10 microinches (10 millionths of an inch), it doesn't even raise the head enough for a human hair to slide between the bottom of the head and disk surface.

While that's mostly an advantage, since we want to keep the head as close to the surface as possible without touching it, it does preclude the use of a floppy-disk-type flexible backing for the magnetic coating; it simply wouldn't do to have a floppy disk spinning at 3600 rpm, waving around and flapping into the head a few microinches above it.

The key element of a hard disk, then, is a magnetic head gliding a few microinches above a very rigid disk surface. Because the head must be kept extremely close to the surface, it becomes imperative to post a no-trespassing sign to particles that might otherwise lodge between the head and the disk surface, lest the glider crash into the particle and land on the disk, taking a large byte out of your data. That, in turn, requires sealing the entire head-disk assembly (HDA) and microfiltering the air within the chamber. The requirement to keep fine junk out of the HDA necessitates that that part of a hard disk always be assembled in a clean room, a factor that permanently raises the price floor of hard disks.

With its clean-room assembly, so much of the cost of a hard disk comes in manufacturing the HDA that it makes sense to squeeze as much in there as possible. And just as adding a second head to a floppy disk drive increases the manufacturing cost a minimal amount, so slipping an extra disk platter or two into the HDA raises the manufacturing cost by only a small increment. Typical 20-megabyte hard disks will have two disk surfaces, or platters, mounted on the same spindle, and four heads—one for each side of each disk—mounted to a single arm and moving in tandem.

The scheme for numbering the sides of a multi-platter hard disk is simply an extension of that for numbering a double-sided drive: The top of the first platter is side 0, its bottom is side 1, the top of the second disk platter is side 2, and its

bottom is side 3. And just as both top-side and bottom-side track 0s on a double-sided drive can collectively be called cylinder 0, so the four track 0s in a two-platter drive can collectively be referred to as cylinder 0.

The amount of data that can be jammed into a single circular track on a hard disk depends on the data encoding scheme used. With current encoding schemes (mostly variants of a technique called MFM, or modified frequency modulation), 8 to 12 kilobytes per track can be reliably read and written.

Although it would be possible to write and read data from a disk in blocks of 8 or 12 kilobytes at a time, that number was historically considered too large to be practical. Disk controllers, both floppy and hard, are therefore designed to read and write only a segment of a track at a time. The particular number of bytes in each segment, more commonly known as a sector, depends on the disk controller hardware and the operating system: The manufacturer designs the disk controller to support several different sector sizes, and the operating system developers choose from among the available sizes. Typically, sector sizes of 128, 256, 512, and 1024 bytes are supported; versions of PC-DOS since 1.0 have used 512-byte sectors exclusively, for both floppy and hard disks.

On a floppy disk, it is possible today to squeeze up to eighteen sectors around a single track and maintain reliability. Conservative engineering, however, led IBM to use only nine sectors per track (and only eight sectors in DOS 1.x); nine sectors per track times forty tracks per side and two sides per disk multiplies out to the now familiar 368,640 bytes per diskette.

With their higher rotational speeds, rigid surfaces, and far more stringent manufacturing tolerances, hard disks can hold both more tracks per side and more sectors per track. The most dramatic improvement comes from cramming in far more tracks per side than is possible on a floppy disk: A typical hard disk may have well over 600 tracks per inch.

(As you can imagine, accurately positioning the head over each one of those very thin tracks requires some tight manufacturing tolerances. That's why multiple-platter hard disks are so popular: It costs no more to move six heads at a time across three platters than it does to move two heads across a single platter.)

Hard disks can also support more sectors in a given track, though here the improvement is not nearly as dramatic: The typical hard disk might have 17 sectors per track, while disks using an encoding scheme called Run-Length Limited Coding, or RLL might have 25 or so. Some of the high capacity disks are using an enhancement of RLL, called ARLL (Advanced Run-Length Limited Coding), which allows 33 sectors or so per track.

Higher data capacity is one of the two major advantages hard disks have over floppies; the other is speed.

While dividing the track up into sectors ameliorates certain problems—there's less data to buffer, for one thing—it creates a new problem: Now we not only need to specify a side and track to find a given piece of data, we also need to indicate what sector number within the specified track we want.

If you read advertisements or spec sheets for hard disks, you're likely to encounter a number of terms used to measure disk performance. In particular, the expressions seek time, access time, latency, and transfer rate keep popping up in any discussion of disk performance. The Norton Utilities Calibrate program measures many of these performance parameters.

Seek Time

Seek time is simply the length of time the disk head takes to move from whatever track it happens to be on, to whatever track you want to read. Since that obviously varies each time you read—depending upon where the head was, and how far it has to go to get to the desired track—there are a couple of sub-species of seek time, the most important of which is track-to-track seek time.

Track-to-Track Seek Time

Track-to-track seek time is the length of time required to move from one track to an adjacent track. Track-to-track seek times in AT-class disk drives are generally on the order of 8 to 10 milliseconds. Seek times for floppy disks are many times greater than those for hard disks.

Access Time

Access time is the time required for the head to seek to (i.e., move to) the track that holds the data you want. Clearly, if you left the disk head on track 3 after your last read and you now want to read data from track 4, the access time would be equal to the track-to-track seek time. But how often will you be lucky enough to have the next track you want to read adjacent to the track you just read? (Actually, more often than you might think. Especially if you're a regular user of Speed Disk.)

Average Access Time

Average access time is a measure of the time it takes, on average, to move the head from the current position to the track you want to read. At first glance, it might seem that average access time would be equal to the time required to seek across half the number of tracks on the drive. That would be true, only if the head were always left at the edge of the disk. But after a disk operation, the head may be left anywhere over the disk—the optimal situation would be when it was left in the center of the disk, since then the next access would, on average, be across half of a half of the total number of tracks, or one-fourth the total.

If you're thinking that the average access time must be somewhere between the time required to seek across one-fourth and one-half the total number of tracks, you're right on target. As it turns out, it can be shown that the average time to access any arbitrary track is equal to the time required to seek across one-third of the tracks. (Note, by the way, that this measurement of average access time is independent of the optimizations that can be performed by an efficient operating system, which will try to organize data in such a way that data to be read sequentially will be stored on sequential tracks. That means that in real life, much of the time we simply have to move a single track.)

Two requirements must be satisfied before the disk head can begin reading data: First, it must seek to the required track; second, the particular sector of the track that is going to be read (or the sector that will be read first, for multiple-sector reads) must spin around under the head. As we've just learned, the average time required for the head to get to the target track is called the average access time. Once at the right track, the time the head spends dawdling until the right sector comes around is called latency.

Average Latency

Average latency is the time required for the disk to make half a revolution. Hard disks spin at 3600 rpm, or one revolution every 16.67 milliseconds, so the average latency is approximately 8.3 milliseconds. Since all hard disks spin at the same speed, this figure doesn't vary from drive to drive, so latency time won't help you compare one hard disk with another. But it's interesting to note that floppies, spinning at 300 or 360 rpm, have an average latency of more than 100 milliseconds, more than ten times greater than the average latency for hard disks.

Transfer Rate

Transfer rate is the speed at which bits are read off the disk—that is, the speed at which they can be transferred from the disk to the computer. Transfer rate depends both on how fast the disk is spinning—you can't read a bit until you get to it—and how densely bits are packed around the track. Here again, hard disks dramatically outperform floppies, both because of the hard disks' higher rotational speed (3600 rpm versus 300) and because of their higher bit-packing density.

Most hard disks today have a transfer rate of 5 megabits per second; with the phasing in of a new hard disk interface standard, known as ESDI, or Enhanced Small Device Interface, transfer rates are in the process of taking a quantum jump to 10 and even 15 megabits per second.

Sector Addressing

Earlier we mentioned an obscure specification of hard disks called the latency time—the amount of time spent waiting for the data we want to rotate around under the head, after we have seeked to the track we want. What we're really waiting for is the particular sector we want to come flying by. The question is: How do we recognize that sector when we see it?

Early floppy disks took a brute-force approach to sector identification. Holes were punched at regular intervals around the circumference of the disk, and each hole, which could be mechanically sensed, marked the beginning of a sector. This method was called hard sectoring.

Apart from being rather inelegant, hard sectoring didn't adapt well to high-performance drives. A less primitive means of sector identification therefore was developed, one that involved encoding each sector's address into the sector's data. This approach came to be known as soft sectoring.

So far, we've been describing strictly physical characteristics of a disk drive: the number of tracks and sides, the size of each sector, access times, and so forth. All of these are characteristics that are determined by the hardware—by the design of the drive itself and that of the controller.

As we've seen, before we can read a sector, we have to put some kind of identifying marker on it. With hard sectoring, holes are punched in the disk, and we use those as a reference in finding each sector. With soft sectoring, before the disk is first used, the sector address of each sector is actually written into the sector itself. (This address stamping takes place during the formatting of the disk.) The address goes into a kind of preamble to the sector data proper. Along with the sector address in the preamble, some special synchronization bytes are written; these are a unique sequence of bytes that appear at the head of each preamble, letting the disk controller know that it's about to read a sector address. And just to make the story complete, we should mention the gap bytes—filler bytes that are placed between sectors to create a timing tolerance for the reading of each sector.

If this sector preamble is destroyed or difficult to read, the error message, "Sector Not Found" will be displayed and the data in the sector will be lost. The Calibrate program will refresh this sector preamble to prevent this type of data loss.

Physical Formatting

Writing the sector addresses, sync bytes, gap bytes, and a few other miscellaneous chunks of data into the sector preamble is called hard, physical, or low-level formatting, because the job can be done only by hardware in the disk controller. During hard formatting, software in the host machine tells the disk controller to format a track, picks one of the available sector sizes, and specifies a few other parameters; from then on, it's the job of the controller itself to execute the format.

The physical formatting must be done before any soft-sectored disk can be used for data storage. A second, separate process, called logical formatting (which we'll talk about in a minute), must also take place before a disk is ready to store data.

What's potentially confusing about the physical and logical formatting processes is the fact that DOS's `FORMAT` or the Norton Safe Format command carries out both aspects of the job for floppy disks, but only the logical formatting for hard disks. When you format a floppy disk, the format command first performs a physical (hard) format on the disk, followed by a logical format; when you format a hard disk, it skips the physical formatting.

Let's look at a situation when you might want to physically reformat your hard disk.

Must You Interleave Me?

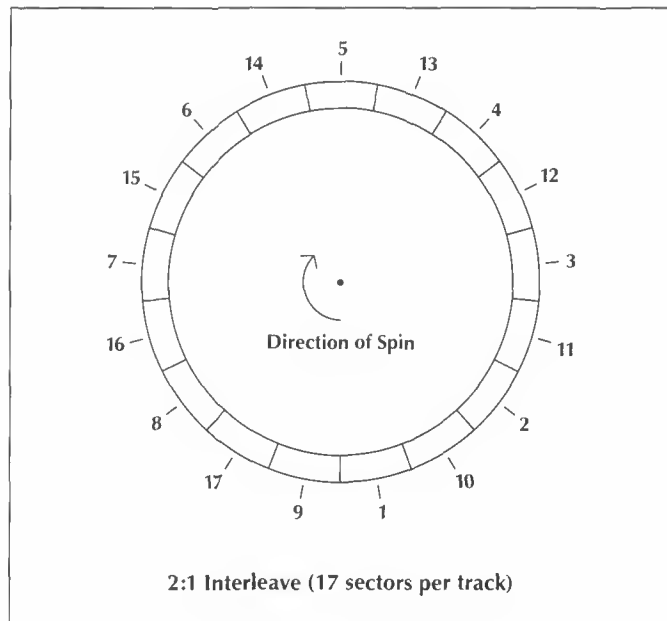
The major job of a hard format is to write the address of each sector into the preamble of sector data. You may have assumed that sectors are numbered sequentially around the track, and in point of fact, many times they are. But there's no law that says they must be, and there are good reasons not to number sectors sequentially.

Suppose you have a computer whose performance is perfectly matched to your hard disk: As data comes flowing out of the disk controller, the computer always manages to jam the data into memory, getting back to the controller just in time to grab the next batch of data that's been read.

A lovely situation. But in real life, things seldom work out quite so copacetically. Frequently, once a disk has seeked to the track it needs, it can start spitting out data faster than the computer can receive it. What happens is this: The controller reads a sector of data, and the computer stores it away. The computer turns back to the controller, ready for more data. But by now, the next sequential sector is history; the third sequential sector is about to come skimming under the heads. Now it's the computer's turn to wait, while the disk makes nearly a full revolution and the beginning of sector 2 spins around again.

This is about the worst match possible, since it yields only a single sector of data for each revolution of the disk. So let's get tricky. Since the sector addresses are simply data—a special kind of data, but data nonetheless—we can change them: We'll just label every other sector sequentially around the track.

Figure 1-1



Interleave

Proceeding around the track, we'll start with sector 1, then we'll skip a sector, then we'll have sector 2, then we'll skip, then sector 3, and so on. When we get back to the beginning, sector 10 will be between sectors 1 and 2, sector 11 between 2 and 4, and so on to sector 9, which would be between sectors 17 and 1 (on a 17-sector disk). We've just changed the interleave factor from 1 to 2, and now when we read sequential sectors from the disk, we'll have twice as much time to process a sector before the next one comes flying by.

With an interleave factor of 2, instead of needing seventeen revolutions to get all the data on a seventeen-sector track, we'll require just two revolutions. That's not as good as getting the whole track read in a single revolution, but it's far better than what we had with the original one-to-one interleave.

Changing the interleave factor is one way we can tune the performance of a hard disk to that of its host computer. But beware: As we've just seen (with the initial example of a disk with no interleaving), when you lower the interleave factor to the point where the computer can no longer keep up with the disk, performance suddenly plummets.

The Calibrate program calculates the optimal interleave factor for your computer system and will also automatically adjust your disk to this new optimal interleave without requiring you to make a backup up your disk first.

THE SOFTWARE

Getting Logical

So far, we've been dealing with physical characteristics of the disk hardware—the number of tracks or cylinders per platter, the number of platters, the sector size in bytes, the number of sectors, the interleave factor, and so forth. These characteristics are either immutable elements of the hardware, as in the case of the number of platters, or factors set by the hardware according to instructions from the operating system (and thereafter seldom altered). There is a whole other group of disk characteristics, however, that has to do with how the operating system organizes and finds data on the disk.

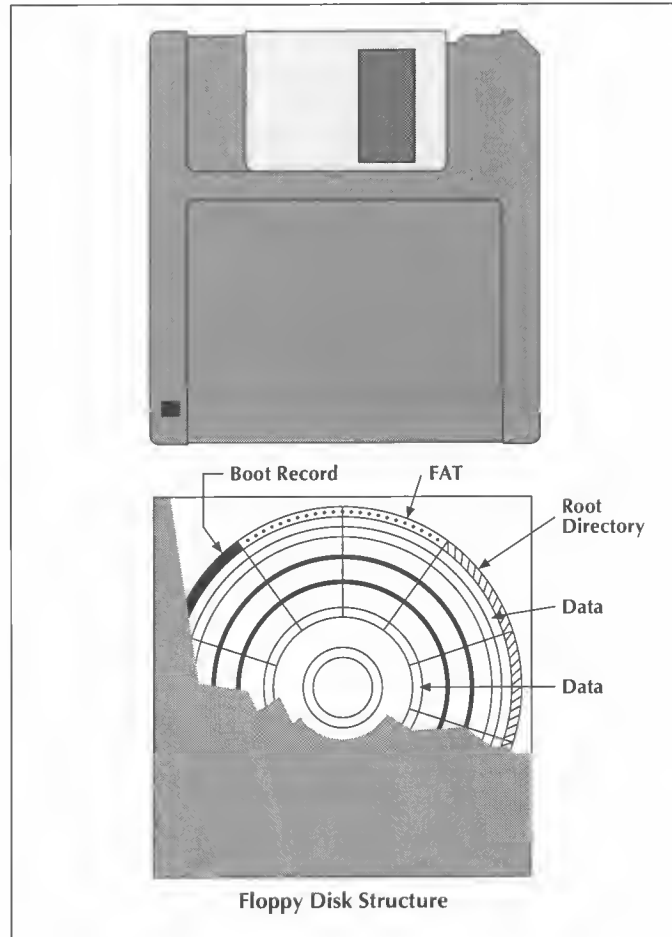
For speed and efficiency in accessing particular bytes out of a vast sea of 20, 40, or many more megabytes, it is clearly necessary for the operating system to construct some directories and indexes telling it what's where, what parts of the disk are spoken for, what parts are free, and even what parts should never be used because of physical damage. The way such information is organized on the disk is called the logical format of the disk, and the process of writing the various directories and indexes that support this organization is called logical formatting. As mentioned earlier, when you format a hard disk using the DOS FORMAT command (or our Safe Format program), all you are really doing is a logical format; the physical format has already been done. With a floppy disk, on the other hand, the DOS FORMAT command first performs a physical format, then automatically follows with a logical format. Safe Format, on the other hand, will only perform a physical format if it needs to.

No matter what kind of disk we use—hard or floppy, fixed media or replaceable, 360K or 60-megabyte—DOS always uses the same logical format, which organizes the disk into four main areas: the boot record, the File Allocation Table (FAT), the root directory, and the data area. (On a hard disk that can be partitioned among different operating systems there is yet a fifth area called a Partition Table; we'll take a look at the Partition Table shortly.)

The Boot Record

The DOS boot record always occupies the first sector of the first track of the first disk side—sector 1, track 0, side 0. (Actually, this is strictly true only of floppy disks, because hard disks reserve the very first sector for the Partition Table.) The boot record does just what its name implies: It lets the computer pull itself by its bootstraps, from empty-minded idiot to efficient manager, by reading in a very short program—the boot code—that in turn reads in the rest of the operating system.

Figure 1-2

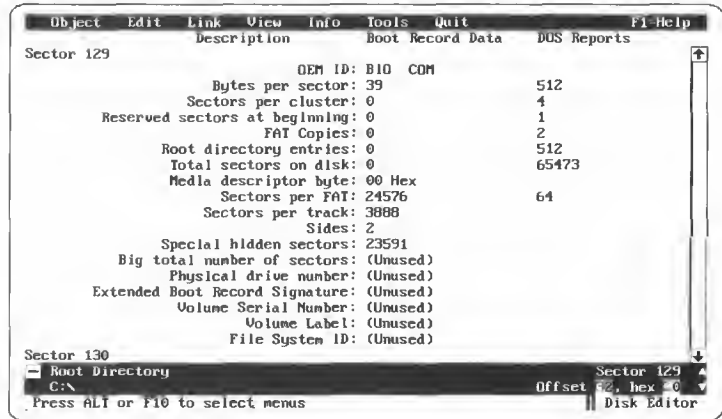


Disk Structure

In addition to the small boot program, the boot record contains another item that's vital to the operating system: a list of key characteristics about the disk. Among the items included in this list are the number of bytes per sector, the total number of sectors on the disk, the number of sectors per track, and the number of heads. Because of the importance of this table, the boot record is written to all disks during logical formatting, even floppy disks that don't contain the three system files necessary to make a self-booting, or system, disk.

The second and third areas that DOS sets up during logical formatting—the File Allocation Table and the root directory—are used together to keep track of where each file is stored on the disk, which sectors are currently in use storing files, and which sectors are available.

Figure 1-3



Boot Record

Clusters

Once a disk has been formatted, the smallest chunk of data that the disk controller is physically able to read or write is a single sector, which, for all DOS disks so far, is 512 bytes. DOS could (and in the case of high-density 1.2 megabyte floppies, actually does) keep track of the status of each individual sector—whether it's in use, free, or shouldn't be used because of damage. But the overhead involved in keeping track of disk space on a sector-by-sector basis is pretty high on a large-capacity disk, so DOS deals instead with multi-sector units called clusters.

Before we look any closer at clusters and how DOS keeps track of files on a disk, it might be a good idea to take a look at some of the problems DOS is going to encounter when we ask it to create, expand, contract, and delete our files.

Files in Flux

The basic problem with managing computer files is that they're constantly changing (of course, the ability to change computer files easily is also what makes them so attractive). But consider: Suppose we have a file that's 2250 bytes long, and we're writing it to a fresh, unused, 1.2-megabyte floppy disk. The smallest number of bytes that can be read or written from a disk is a single sector, and that is indeed the minimum number of bytes DOS will read or write on a 1.2-megabyte floppy. We'll need 5 sectors to hold our 2250-byte file. The first four sectors will hold 2048 bytes, which means we'll need only 202 bytes of the last sector. Since we can't write a partial sector, DOS will reserve all of the fifth sector for our file, letting the remainder of the last sector go to waste. Our new file will occupy the first five sectors of the disk.

Now let's write a second file that's, say, seven sectors long. It will occupy the next seven sectors of the data area. Now comes the fun: Suppose we want to add 460 bytes to our first file. As you'll recall, only 202 bytes of the last 512-byte sector was in use, so we still have room for 310 bytes there. But that leaves 150 bytes (460 minus the 310 bytes we added at the end of the last sector); where shall we put them?

We can't simply put them in the first sector after our five-sector file, because that sector is already in use by the next file (the seven-sector file). We could move that next file up a sector, but that would take a while. And, besides, consider how inefficient it would be if the next one megabyte of sectors were already taken by files: We'd have to move a megabyte of data just to write a lousy 150 bytes. Not a good solution.

The File Allocation Table

A much better solution, the one that DOS uses, is to maintain a table of the status of all sectors. This important table, called the File Allocation Table, or FAT, will tell us whether a given sector is free or in use by a file. Now when we need to expand a file, we just look in the FAT to find the next free sector and reserve it for our file. In this example, the first 5 + 7 sectors are taken, so the first free sector is located some 13 sectors from the beginning of the data area. Our file now occupies the first five sectors of the data area, plus the 13th sector of the data area. So our file is fragmented

now. The beginning portion is stored in contiguous sectors at the beginning of the disk, while the ending portion is elsewhere. This is not a problem, however, provided we can figure out a way to link this ending piece of the file with the beginning piece, so that DOS knows where to find everything.

For the sake of clarity, we just said that DOS keeps a table, called the FAT, that records the status of all sectors on the disk. That's not exactly true. For a 32-megabyte disk, a table that kept track of each sector would be 65,535 entries long (32 megabytes divided by 512 bytes per sector). Since for fast access the FAT is stored in memory while the disk is in use, large disks would eat up memory pretty fast; besides, it would take a while to scan through a 128K FAT (64K entries, times 2 bytes per entry). For that reason, DOS groups a number of contiguous sectors together, and deals with them as a unit called a cluster. It's actually clusters, not sectors, that DOS keeps track of in the File Allocation Table.

Since clusters are a fabrication of DOS, DOS can define a cluster to consist of as many sectors as it wants. Rather than settling on a single cluster size, however, DOS varies the size to suit the medium; the cluster size of a given disk is set during logical formatting and does not change thereafter (unless the disk is reformatted). The 30-megabyte AT hard disk is given a cluster size of 4 sectors, or 2048 bytes. The 1.2-megabyte floppy, on the other hand, is formatted with a cluster size of one sector: 512 bytes. That small cluster size reflects a tradeoff of speed for optimal use of available space. It takes longer to deal with disks with smaller clusters, since more entries must be dealt with in the FAT, more read and write operations are required, and so on. On the other hand, the smaller the cluster size, the less space is wasted at the end of each file.

A moment ago, we were looking at a file that was 2250 bytes long and occupied five sectors on a high-capacity disk. Since each cluster on a high-capacity disk consists of a single sector, we only wasted the bytes at the end of the last sector—er, cluster—which amounted to about 200 bytes. But suppose we moved that same file to a 30-megabyte hard disk,

formatted with a cluster size of four sectors? DOS uses the FAT to keep track of space on a cluster-by-cluster basis; since a 32-megabyte disk will probably have a four-sector (2048 byte) cluster size (in DOS 3.x), the minimum amount of space DOS can allocate on that disk is 2048 bytes. Our 2250-byte file will take only two clusters when we move it, but since the clusters are now 2048 bytes apiece, the number of wasted bytes at the end of the last cluster has jumped from around 200 to more than 1800.

That unusable space at the end of a cluster is called slack, and it's the reason files sometimes seem to shrink when you move them from a hard disk to a floppy and expand when you move them in the other direction. We accept the greater slack intrinsic to larger cluster sizes in return for the better performance we get with larger clusters.

Beefing Up the FAT

Back to the FAT. So far, we've said that DOS maintains a table at the beginning of each disk called a File Allocation Table, or FAT. We know that the FAT is used to keep track of which clusters are free, and which are in use and by what file. We also know that a cluster is simply a convenient grouping of one or more sectors, invented to cut down the amount of bookkeeping necessary to keep track of space usage on a hard disk. What we haven't examined so far is how DOS actually stores that information in the FAT.

The layout of a FAT is simplicity itself. Two bytes (for DOS 3 and later) are allocated for each cluster on the disk. The sequence number of each two-byte entry corresponds directly with the equivalently numbered cluster in the data area. Whoops—we need to back up again: We haven't learned how DOS numbers clusters yet.

Three Ways of Counting

Back when we looked at the physical layout of a disk, we found that the disk controller refers to a particular location using a three-coordinate system: the side number (0 or 1 for a floppy), the track number (0 to 39 for a double-sided, double-density floppy), and the sector number (1 to 9 for a 360K floppy). Since the hardware deals with disks in this three-coordinate system, it's necessary that, at some level, the PC be able to deal with the disk using the same system—

which is why all the built-in ROM-BIOS calls specify disk locations via side, track, and sector. For the operating system, however, this three-coordinate system is awkward, what with the number of tracks, sectors, and sides all changing from one disk to the next. DOS therefore locates data on a disk by a one-dimensional sector numbering scheme; DOS simply numbers all the sectors sequentially, starting with sector 1 of side 0, track 0, proceeding through all the sectors on track 0, then on to the sectors on side 1, track 0. All the sectors in a cylinder are numbered first before DOS moves on to the next cylinder; that minimizes head movement, and thus access time, when sectors are read sequentially. The continuous sector-numbering scheme that DOS uses is called logical sector numbering; unlike physical sector numbers, which start at 1, logical sector numbers start at 0.

Don't be confused by the statement that DOS numbers the disk sectors sequentially; there's no physical formatting going on here, no addresses actually written on the disk. What's occurring is simply a translation between the way DOS refers to a given sector and the way the BIOS and disk controller refer to it. Think of it as the difference between calling it "sector one" in English and "premier secteur" in French.

Once we've got DOS's sector numbering scheme down pat, the cluster numbering scheme follows pretty quickly. First, we skip over all the sectors occupied by the boot record (always one sector), the FAT (the size of which varies with each disk format but is constant within each format), and the root directory (which we haven't discussed yet). These three areas, which are always set up during logical formatting, are all considered part of the system area and are excluded from the cluster numbering scheme.

All the sectors beyond the system area, starting with the first sector after the root directory, are considered part of the data area and are included in the cluster numbering scheme. On a disk with a cluster size of four sectors, the first four sectors of the data area will be called cluster 2 (that's right, cluster numbering starts at 2, not 0 or 1), the next four sectors will be cluster 3, the next four will be cluster 4, and so on to the end of the disk. A 30-megabyte disk, with a cluster size of four

sectors, would have about 15230 clusters of data area; the last cluster number would be 15231, one larger than the number of clusters, since cluster numbering starts at 2.

Once More, Back to the FAT

With that out of the way, let's return to the FAT once more. We took this last detour when we observed that the FAT contains information about the status of each cluster on the disk, and that there's one FAT entry for each disk cluster. The first two FAT entries are reserved for special information. The third FAT entry holds information about the first cluster (which, to keep us on our toes, is numbered cluster 2). The next entry holds information about the second cluster (cluster 3), and so on to the end, so that the last FAT entry tells us the status of the very last cluster on the disk. (Strictly speaking, some of the details we're describing now pertain to all floppy disks but only to hard disks with a single partition. We'll get to partitions at the end of this section; their presence doesn't change the basic workings of a disk.)

Let's see what information the FAT entry needs to give us. First, since even the best disks can have a few bad sectors, the FAT entry for a given cluster could tell DOS if there's a bad sector anywhere in the cluster; that way, DOS would know never to allocate that cluster to a file. And indeed, that's one of the pieces of information that DOS records when it creates the FAT during logical formatting. After DOS has finished formatting a disk—writing the boot record and constructing a fresh FAT and root directory—it attempts to read every sector on the disk; clusters that contain sectors that can't be properly read are marked as bad in the FAT. This process is called bad sector mapping, and it's very important to the integrity of our data.

Incidentally, you might be wondering what data DOS could be trying to read off a freshly formatted and supposedly empty floppy disk. A good question, the answer to which is that a "blank" floppy disk is not empty; it's only devoid of data useful to us. When DOS or Safe Format formats a floppy, it writes a special byte (F6 hex) across all of the data area. Later in the formatting process, it tries to read that byte from each sector.

There may be some sectors that the disk controller is unable to read; the controller in this case passes an error message back to DOS, and DOS marks the cluster containing that sector as bad. A few other sectors may be read okay by the controller but have incorrect cyclic redundancy check bytes, indicating there has been an error in reading the data.

Cyclic Redundancy Check

You may have heard it called the CRC. When a sector is written, a special checksum value, called the cyclic redundancy check, is calculated, based on the value of all the bytes written to that sector. This value (also known as a CRC) is written to a special location on the disk immediately following the sector data. When the sector is later read off the disk, its checksum, or CRC, bytes are also read. The same calculation that determined the CRC bytes that were written on the disk following the sector data is performed again, this time on the data that was just read. The CRC calculated from the data just read is compared to the CRC that was calculated and written to the disk. If the two values don't match, it means the data read is not exactly the same as the data that was written, and therefore that a read error has occurred. The Norton Disk Doctor program marks the cluster containing that sector as bad so that area of the disk won't be used again.

By the way, you may be surprised to learn that when you tell DOS to verify a disk copy—using either the /V switch with the COPY command, or the command SET VERIFY ON, DOS does not compare the source data byte-for-byte with the destination data; it simply rereads the destination file and verifies that there are no CRC errors or any other type of read errors. If the CRC checks out, chances are good the data is correct.

But enough of these tangents; let's finish our tour of the FAT. So far, we have seen that we need to keep two facts about each cluster in its corresponding FAT entry—whether the cluster is free or in use by a file and whether the cluster is bad. And if files didn't randomly grow, shrink, or get deleted, that's about all we'd need to keep in the FAT. But remember earlier when we were writing a short, five-cluster file at the beginning of a disk, followed by a second, unrelated file? Then when we tried to add some data to our first file, we

couldn't grab the next sequential cluster, because it was already in use by the second file; we had to grab a cluster somewhere further down the road. That presents a problem: When DOS tries to read the first file, how will it know where to find that other cluster, the one we wrote further down the disk?

The answer is simplicity itself. We wrote the first five clusters of the file into the first five contiguous clusters on the disk, then skipped some clusters and wrote the file's sixth cluster; why not have the FAT entry for the last contiguous cluster we wrote—the fifth cluster—point around those skipped clusters to the sixth cluster of the file? If the file was written into clusters 2 through 6, and then we added cluster 14, we could write a 14 into the FAT entry for cluster 6. That 14 in cluster 6's FAT entry would say: "You will find the next portion of this file at cluster location 14 on the disk." For consistency, we would make each FAT entry that's in use by a file point to the next FAT entry in the file, so we would have a chain of FAT entries, each entry pointing to the next. And since the sequence number of each FAT entry is the same as the cluster it is reporting on, to locate each cluster in the file, we would just follow its chain in the FAT. This is exactly the procedure that DOS follows.

Figure 1-4

Object	Edit	Link	View	Info	Tools	Quit	FI-Help
Sector 129							
16969	16973	20297	8224	20291	10061	0	0
0	0	0	24576	3888	2	23591	0
16969	17485	21327	8224	20291	10061	0	0
0	0	0	24576	3888	14	30632	0
20545	21328	8224	8224	8224	4128	0	0
0	0	0	21262	5169	29	0	0
20292	8275	8224	8224	8224	4128	0	0
0	0	0	21307	5169	43	0	0
20302	21586	20047	8224	8224	4128	0	0
0	0	0	34135	5176	44	0	0
20301	21333	8261	8224	8224	4128	0	0
0	0	0	25322	5399	49	0	0
21825	20308	22597	17221	16706	8276	0	0
0	0	0	38922	5717	3968	302	0
18775	22350	21071	8260	8224	4128	0	0
0	0	0	31826	5425	51	0	0
14387	19766	22593	8224	8224	4128	0	0
0	0	0	24106	5244	54	0	0
16717	19529	8224	8224	16706	8276	0	0
0	0	0	29345	5742	33	108	0

Root Directory Sector 129
 Drive C: Cluster 3,493,449,747, hex D039D013
 Press ALT or F10 to select menus Disk Editor

Cluster Chains in the File Allocation Table

If you've followed us to this point, you should have a good understanding of the basics of DOS's disk file structure. We just need to fill in a couple of simple details. Such as: Now that we know how to find all the clusters in a file by tracing through its chain in the FAT, how do we locate the beginning of that chain? In our example, the first cluster of our six-cluster file was also the first cluster of the disk's data area—but only because we said it was. How are you going to find that starting cluster tomorrow? And do you remember where we said the second file starts?

Obviously, we need to keep a list somewhere of the starting FAT entry for each file. If we know where the first FAT entry is, we also know where the first file cluster is (they're numbered the same, remember, with each FAT entry reporting on its equivalently numbered cluster), and since each FAT entry points to the next one in the chain, we can find all the clusters in our file.

In addition to the list of starting cluster numbers, which enables us to find the first cluster in each file, we also need a means to find the last cluster in each file. We could put that information in the same place as the starting-cluster information, by listing either the last cluster number or the file's size. Alternatively, we could use the FAT itself, placing a special byte into the FAT entry that corresponds to the last cluster of each file.

In a way, DOS does both: The file directory (which we'll come to presently) records both the length of each file and its starting cluster number, and the last FAT entry of each file is indeed specially marked.

Our discussion of the all-important FAT is now nearly complete. We know that the FAT is a table of two-byte entries (starting with DOS 3.xx), with one entry corresponding to each cluster on the disk. (Earlier versions of DOS used 1.5 bytes for each FAT entry. The practical significance of this is discussed at the end of this section.) Each entry tells us whether its associated cluster is available, defective, or

already in use by a file. If the cluster is in use, the entry either points to the next cluster/FAT entry of the file, or tells us the cluster is the last cluster of the file. So a FAT entry can contain one of four values:

Value (hex)	Meaning
0000	This cluster is available
0002-FFEF	Cluster in use by file (the number points to the next cluster in the file)
FFF0-FFF6	Reserved; not used
FFF7	Bad cluster; do not use
FFF8-FFFF	This is the last cluster in a file

And we know that there is information in a directory that points to the starting FAT entry for each file.

The Root Directory

The directory—actually the root, or main, directory—is the third and last part of the system area of any disk formatted by DOS. If you've ever looked at a directory using Disk Editor, you've probably discovered that each directory entry holds a good deal more information than the starting cluster number and size of each file.

Clearly, in addition to the size and starting cluster numbers, each directory entry needs to list the filename associated with the starting cluster number.

Looking at it another way, we could say that there is one directory entry for each file on the disk, and that a file's directory contains a pointer to its first entry in the FAT chain, i.e., its first cluster. In addition, each 32-byte directory entry contains fields for the time and date, a one-byte field of file attributes, and ten bytes that are reserved by DOS. So the format of each directory entry looks like this:

Figure 1-5

Object	Edit	Link	View	Info	Tools	Quit	FI Help
Name	Ext	Size	Date	Time	Cluster	Arc	R/O Sys Hld Dir Vol
APPS		0	1-17-90	10:24 am	29		Dir
DOS		0	1-17-90	10:25 am	43		Dir
NORTON		0	1-24-90	4:42 pm	44		Dir
MOUSE		0	8-23-90	12:23 pm	49		Dir
AUTOEXEC	BAT	302	2-21-91	7:00 pm	3968	Arc	
WINWORD		0	9-17-90	3:34 pm	51		Dir
386MAX		0	3-28-90	11:49 am	54		Dir
NAIL	BAT	108	3-14-91	2:21 pm	33	Arc	
TEST		0	3-20-91	10:01 pm	10496		Dir
ANNUAL	RPT	0	4-12-91	5:58 pm	7799		Dir
NAV		0	11-29-90	5:51 pm	66		Dir
NAVU		0	11-29-90	5:54 pm	67		Dir
NDW		0	10-11-90	2:36 pm	68		Dir
WINDOWS		0	9-17-90	3:52 pm	69		Dir
Sector 130							
REVIEWS	EMP	0	4-12-91	6:01 pm	7801		Dir
SYSTINFO		0	10-17-90	5:34 pm	74		Dir
NET		0	8-20-90	1:05 pm	75		Dir
HINEN	SYS	11304	5-01-90	3:00 am	6564	Arc	
FINANCES		0	4-12-91	6:03 pm	7805		Dir

Root Directory Sector 130
C:\ Offset 128, hex 00
Press ALT or F10 to select menus Disk Editor

Directory Entry

Description	Size (Bytes)	Format
Filename	8	ASCII characters
Extension	3	ASCII characters
Attributes	1	Each bit represents an attribute
Reserved	10	Unused (so far)
Time	2	Word, coded
Date	2	Word, coded
Starting FAT entry	2	Word
File size	4	Long integer

The attributes represent one of several special properties that can be applied to each file, such as read-only, hidden, system, volume label, subdirectory, and archive. The attributes are bit-encoded; a given attribute is set if its associated bit is a 1.

The volume label attribute presents an unusual situation. No file actually exists for a directory entry with this attribute. Only one directory entry in a disk should have the volume label attribute, and that entry should be in the disk's root directory. The label is stored in the filename and extension fields of the directory entry, which in this case are treated as a single field.

Before we get too wrapped up in the innards of directories, let's take a moment to distinguish between the root directory and subdirectories. So far, we've been talking about the root directory as if it were the only directory on the disk, and indeed, the root directory is special in a couple of ways. First, the root directory is the only directory that is not in the disk's data area: The root directory is the third element of the system area, and it's always located immediately following the FAT. The size and location of the root directory are fixed; they are established during logical formatting and cannot be changed afterward. Its fixed size also differentiates the root directory from subdirectories, which can be created, grow, shrink, and be deleted as necessary.

The size of the root directory varies with the type of disk. On a 360K floppy, format creates a root directory with room for 112 entries; on most hard disks, format creates a root directory that can hold 512 entries. That means, for example, that if you want more than 512 files on a 32-megabyte hard disk, you'll have to create some subdirectories.

Any entry in the root directory can refer to either a file or a subdirectory. Subdirectories are hybrids. They are assigned space exactly as though they were files. They can grow, expand, and be deleted like a file. But, rather than holding our data, they hold other filenames. You might say that in form, a subdirectory is just like any other file, and in function it's exactly like the root directory. Because subdirectories are stored in the data area, they can be assigned space on an as-needed basis.

A while ago, we said each directory entry contained, among other items, the filename, file size, date and time of file creation (or last modification), and the starting cluster number (FAT entry) of the file. Let's take a closer look at the filename field of a directory entry.

As you might expect, the filename field is an eleven-byte area, divided into an eight-byte name field and a three-byte extension field. Filenames of fewer than eight characters are padded to the right with blanks; immediately following the eight-byte filename is the extension, padded with spaces to

three bytes. (The dot isn't stored; it's assumed to be between the eighth and ninth characters.) If you use the Disk Editor to look at a filename, you will see it is always stored in CAPITAL letters; if you ever change the filename in a directory by means of the Disk Editor be sure to use CAPITAL letters, because DOS will choke on lowercase letters in a filename.

Just as a FAT entry can either be a pointer to the next entry in the chain, or one of three special codes, so the first byte of the filename field can either be the first letter of the filename, or one of three special codes.

A 0 as the first byte of the filename indicates a completely unused directory entry. Using a 0 to indicate virgin directory entries enables DOS to know when it has reached the end of active directory entries, without searching to the end of the directory. (This convention was not implemented in DOS 1.x.)

A period character as the first byte of the filename indicates that the entry is reserved by DOS, for use in navigating around the directory structure.

Last, but definitely not least, the first byte of the filename may contain a lowercase Greek sigma character. This special marker (whose ASCII value is 229 decimal, E5 hex) indicates that the file has been erased. UnErase represents the sigma character as a question mark, to indicate that it represents an erased and now unknown character.

Fortunately for us and for the Norton Utilities UnErase program, DOS is pretty lazy when it comes to erasing a file. Rather than erasing the actual file data, it simply marks the first byte of the filename with an E5 hex marker, to indicate that the file has been erased, then clears (zeros) the file's entries in the FAT. (Recall that a value of 0 in the FAT entry indicates that the cluster is available.) Since it doesn't erase the actual data, or even the starting cluster number in the directory, we can find and recover the first cluster very easily, provided it hasn't been overwritten by another file.

Using knowledge about the structure of the disk, directory, and FAT, the Norton Utilities UnErase program can often recover the rest of the clusters of an erased file automatically; the utility even knows how many clusters to look for, since the erased file's length in the directory is not overwritten. Sometimes the file is so badly fragmented that UnErase either can't locate the right clusters or finds the right clusters but puts them together in the wrong order; that's the sort of problem that the UnErase program's Manual UnErase option, with its more sophisticated capabilities, is designed to handle. Working with what you know about your file and what UnErase knows about your disk, you can frequently recover even badly fragmented and partially overwritten files.

To be on the safe side, save copies of your root directory and FAT with Image. This program takes a snapshot of your system information and saves it to the disk. When you need to recover a file, UnErase uses this information in order to recover it.

Tying Up the Loose Ends

So far, we've been talking about two main areas the disk is divided into—the system area and the data area—and the three subdivisions of the system area. To wit, we said the system area is divided into a boot record, which is always the first sector of the disk; the FAT, which directly follows the boot record; and the root directory, which immediately follows the FAT.

This description of the system area is 100 percent accurate for floppy disks and other special disks that can't be partitioned for use by multiple operating systems.

High-capacity hard disks are valuable and expensive resources, so, good neighbor that it is, DOS provides a way to share such disks among different operating systems. The trick is to set up multiple partitions, with one partition for each operating system. (It is possible to set up multiple partitions all running the same operating system, but there's little reason to do so, unless you're running a version of DOS up to version 3.3 and you have a disk larger than 32 megabytes, which is the largest capacity disk such DOS versions can handle; Compaq DOS 3.31 and PC-DOS 4.0, among others, can support partitions larger than 32 megabytes.)

Since DOS couldn't get the time of day from a Unix partition, and Unix doesn't know how to read a DOS partition, it follows that there must be some area of the disk common to all partitions. This common area, known as the Partition Table, specifies the disk location and length of each operating system's partition.

The Partition Table is always found in the first sector of any partitionable disk (just as the boot record is found in the first sector of any non-partitionable disk). On a partitionable disk, the Partition Table must be set up before logical formatting. (You may recall that logical formatting is the only type of formatting the DOS `FORMAT` command or Safe Format does on a hard disk; physical formatting of a hard disk is generally done by the manufacturer.) The DOS command `FDISK` sets up the Partition Table. That's all it does, as a matter of fact. Before you can use a hard disk, then, you must first run `FDISK`, then run DOS's `FORMAT` or Safe Format.

If the partitioned disk is also the boot disk, the question arises: With multiple operating systems residing on the same disk, how does the PC know which system to start up at boot time? The answer is: This information is kept in the Partition Table.

Several other miscellaneous housekeeping items are stored in the Partition Table, but only one of them is of any real interest to us. We said that on a non-partitionable disk, the first sector is always the boot record, a short program that loads in the rest of DOS. Since DOS expects to find a boot program in the first sector, partitionable disks oblige by also putting a boot record in the first sector, as part of the Partition Table.

In a case like this, when the boot record (boot program) is part of a Partition Table, it's actually called a master boot block. Now when we boot off our partitionable disk, we'll first read in the code in the master boot block of the Partition Table; that code will in turn find out which partition is active and read the boot code from that partition. So if the DOS partition occupied the entire disk, the master boot code would read in the DOS boot record, which would in turn

read in the rest of DOS. If there were two partitions, and Xenix were active, the master boot code would read in the boot code from the Xenix partition, which, being partial to Xenix, would read in the rest of Xenix.

That's all there is to the Partition Table. The remainder of a partitionable disk is organized exactly as described above, except that now our description of the organization of a DOS disk applies only to the DOS partition. In the typical case, where the entire disk is given over to the DOS partition, the only difference would be that everything is "slid forward" one track, to make way for the Partition Table. (The Partition Table uses only the first sector of track 0, but the remaining sectors on the track are usually skipped anyway.) The DOS boot record would still start at sector 1, but of track 1; the FAT would start at sector 2, track 1, and so on.

Figure 1-6

System	Boot	Starting Location			Ending Location			Relative Sectors	Number of Sectors
		Side	Cylinder	Sector	Side	Cylinder	Sector		
DN	No	0	80	0	83	32	32	538976288	4128
unused	No	0	0	0	0	0	0	351836692	2671
DN	No	0	80	0	83	32	32	538976288	4128
unused	No	0	0	0	0	0	0	351836692	2671

Root Directory: Sector 131
 Drive C: Offset 752, hex 100
 Press ALT or F10 to select menus
 Disk Editor

Partition Table

Speaking of the FAT, let's confuse one more item that we simplified during our discussion. You'll recall that we've been talking about two-byte FAT entries; two bytes being 16 bits, and 16 bits being capable of holding a number as large as 65,535, a two-byte FAT entry should be good for disks with up to 65,536 clusters.

Alas for our discussion, the FAT with two-byte entries is a Johnny-come-lately. Until DOS 3.0, all FAT entries were 1.5 bytes. One and a half bytes, or 12 bits, can express numbers up to only 4095, which is a relatively small number of clusters. Organize a large hard disk with a small number of clusters, and you've got to make each cluster a large number of sectors—which, unless your files are gargantuan, is inefficient.

As of version 3.0, therefore, DOS allows for FATs with two-byte entries on large disks, while continuing to use 1.5-byte FATs on smaller media. Whether its entries are 1.5 or 2 bytes long, the FAT still works the same, so the practical significance of having two different-sized FATs is nil, unless you like doing FAT arithmetic (which you won't need to do if you're using the FAT viewer in the Disk Editor; it does the FAT arithmetic for you).

If you are using the FAT viewer, we should mention one more thing: The FAT is so important, DOS stores two identical copies of it.

Finally, let me take care of one more little item. You've probably heard about the 32-megabyte file size limit imposed by DOS, and you may have wondered where it came from. Let's do a little arithmetic. DOS, you recall, transforms the physical side/track/sector three-dimensional numbering scheme used by the disk controller and the BIOS into a single, sequential, one-dimensional sector number. DOS uses a two-byte word to specify that sector number, so the largest sector number it can handle is 65,535, the largest number that two bytes can express. If we've got 65,536 sectors (DOS logical sector numbering starts at 0, even though physical sector numbering starts at 1), and each sector is 512 bytes, then the maximum number of bytes we can address is 65,536 times 512, or 32 megabytes.

A way around the 32-megabyte limitation is to increase the sector size. With a sector size of 1024, for example, we could address up to 64 megabytes. This is the tack taken by the distributors of some of the immense hard disks now being offered for the PC. Compaq's DOS 3.31 was the first to surpass this limitation by increasing the two-byte number

specifying a DOS sector to four bytes. Previously, the two-byte specification allowed up to 65,536 sectors. Using four bytes, on the other hand, allows up to 4 billion sectors. (What famous astronomer does that remind you of?) Therefore, this version of DOS can handle partitions greater than 32 megabytes. PC-DOS and MS-DOS 4.0 and greater also support partitions larger than 32 megabytes using this same scheme.

THINGS THAT GO BUMP IN THE DISK

Now that we know how a disk should work, we are in a pretty good position to understand the kind of things that can go wrong—and more important—how to avoid or fix them.

Between the directory and FAT, it's pretty easy to imagine some of the things that can go awry. It's possible, for example, for a cluster in the FAT to be marked as in use, yet not be part of any file's allocation chain. Such an anomaly would signal an orphaned cluster (sometimes called a lost cluster). Getting more imaginative, the FAT entry for one cluster could point back to the entry that just pointed forward to this entry, creating a circular chain. Or the allocation chains for two or more entries could both point to the same cluster, meaning a single cluster had been allocated to two different files—a mistake referred to as a cross-linked file.

Such errors can happen either when the moon is full and strange creatures roam the night, or when you turn off your machine while a program still has a file opened or is writing to the disk. Quit your programs before shutting down.

In any case, the The Norton Disk Doctor program can spot, report, and fix logical errors such as these, all with a minimum of intervention from the user. In this way, it's far superior to the DOS CHKDSK program. In fact, CHKDSK will cheerfully mislead you if you run it without the /F switch, by saying that it fixed the errors; don't believe it.

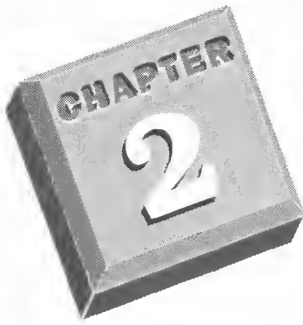
What's more, CHKDSK doesn't check for, and can't fix, physical disk errors, such as sectors that can't be read. Disk Doctor can handle these sorts of problems. The program can reformat bad sectors and write back the old data. In addition, Disk Doctor can detect and fix bad Partition Tables, detect and fix bad boot records, detect and correct problems with the FAT, and check out and correct the root directory.

In a way, Disk Doctor represents a translation of the knowledge in this section into a very powerful program. Disk Doctor is designed to give even a relative novice the power to correct a variety of problems with a minimum of effort. Even though you may not need to know much about how your disks work in order to keep them in top shape, however, a little inside information never hurt anybody. Except maybe a few unlucky souls on Wall Street.

CONCLUSION

If you've come this far, you should have a pretty good understanding of what's going on behind that little red disk light. We haven't discussed everything there is to know about disks, of course, but we have covered all the important concepts and terms you're likely to run into in your day-to-day dealings with your PC and DOS.

If you should decide to test out some of the material you've learned, poke about your disk with the Disk Editor, maybe even discover a few items we didn't have space to mention here, we hope you'll discover that the Norton Utilities are as much fun for exploring your disk as they are useful for recovering data.



Disk Editor

Contents

What It Does	2-3
When to Use It	2-4
How to Use It	2-5
Before You Start	2-5
Running the Program	2-5
Getting Help	2-6
Configuration	2-6
Selecting an Object	2-8
Selecting a File	2-9
Identifying the Disk	2-10
Selecting a Directory	2-11
Selecting a System Area	2-11
Selecting Clusters	2-12
Selecting Logical Sectors as Objects	2-13
Selecting Physical Sectors as Objects	2-14
Linking to Related Objects	2-15
Print As... ..	2-17
Memory Dump	2-18
Displaying Information	2-19
Object Information	2-19
Drive Information	2-20
Map of Object	2-20

Editing Objects	2-21
Selecting a View	2-21
Making Changes	2-23
Undoing Changes	2-24
Saving Changes	2-25
Creating a New Object	2-25
The Hex Viewer	2-26
The Directory Viewer	2-29
The Boot Record Viewer	2-31
The FAT Viewer	2-33
The Partition Table Viewer	2-34
Power Editing Features	2-37
Tools	2-43
Calling DOS Temporarily	2-44
Terminating Disk Editor	2-45
Menu Reference	2-46
Shortcut Keys	2-46
Object Menu	2-46
Edit Menu	2-47
Link Menu	2-48
View Menu	2-49
Info Menu	2-49
Tools Menu	2-50
Quit Menu	2-51
Operational Notes	2-52
Command Line Use	2-52

WHAT IT DOES

The Norton Disk Editor lets you view and edit the entire contents of a diskette or a hard disk. You can even work on disks DOS cannot recognize or access. You can work not only on files and directories, but also on the Partition Table, the Boot Record, and the File Allocation Table. If none of these objects meets your needs, you can treat any group of clusters or sectors as an object. (An object is the entity that you have chosen to view and edit.) So you can, for example, rescue individual sectors from a bad cluster, find the root directory on a trashed disk, or explore the unused portions of the disk. Detectives use Disk Editor to find data that criminals have tried to stash in hidden partitions on their hard disks.

Disk Editor permits five different views of the selected object: Hexadecimal, Directory, FAT, Boot Record, and Partition Table. So you don't have to know the formats of the various types of objects; the disk viewers interpret and format the data for you. The directory, Boot Record, and Partition Table editors even label the fields and prevent you from entering invalid values. You can also view, but not edit, an object in text mode.

You have the option to print in Disk Editor, too. Any selection can be printed in either Hex, Directory, FAT, Boot Sector, or Partition Table format. The default print format will be the same as the current view, but the user can select a format via a set of radio buttons.

Disk Editor offers many additional facilities, including:

- an information facility that displays useful facts about the currently selected object such as its size and location on the disk
- a find facility that lets you search through the object (which might include all the clusters on the disk) for a particular data string
- a windowing facility that lets you view two objects at once
- a compare facility that finds the differences between two objects

- a fill facility that overwrites an area on the disk with a specific value
- a clipboard facility that lets you copy any block of information to any other location
- a write facility that copies the object to another disk as a file or simply as sectors or clusters
- a PRINT AS... feature to print any selection
- MEMORY DUMP option, giving you the ability to view, search, and copy data from conventional memory
- access to the Diagnostic Cylinder
- an undo facility that reverses changes to a sector before they are saved

Disk Editor makes it relatively easy to rescue a troubled disk, or simply to explore the disk for the fun of it.

WHEN TO USE IT

There are situations when only Disk Editor will get things back to normal on a disk. The most common of these situations are described in the “Troubleshooter” section of this manual. A short list of situations where Disk Editor may be the only tool to do the job includes the following:

- Norton Disk Doctor is unable to correct all of the disk problems you are experiencing
- data drops out of or appears unexpectedly in a file, implying FAT or directory problems
- you need to lift data off a bad disk
- you need to patch strings in files
- you want to print a selection
- directory contents are scrambled or unusable
- you want to find out what’s in the unused portion of the disk
- unknown files appear on the disk, and you can’t read them through normal methods
- you want to browse disks formatted with some non-DOS operating systems

HOW TO USE IT

This section explains how to use Disk Editor. However, you should be aware of the following before you get started:

Before You Start

Disk Editor will run faster if you can install it on your hard disk, but if it has not yet been installed, *don't* install it in either of the following cases:

the hard disk is unreliable or inaccessible, or
you are trying to recover lost data from the hard disk.

In the latter case, you might overwrite the very data you are trying to recover with the Norton Utilities; be very careful not to add any data to your hard disk until you have successfully recovered the lost data.

The beginning of the next section explains how to run Disk Editor from a floppy drive in such cases.

Running the Program

To run Disk Editor from the Norton Utilities main menu, follow these steps:

Step 1. Type `NORTON` at the DOS prompt. The Norton Utilities main menu appears.

Step 2. Select Disk Editor in the scrollable list of commands.

Disk Editor tells you it is in read-only mode and tells you how to turn read-only mode off: select `CONFIGURATION` from the Tools menu and deselect the Read Only check box (with the mouse or Spacebar). Then Disk Editor scans the disk and switches to the directory viewer for the current directory.

Getting Help

To get command line help, at the DOS prompt, type:

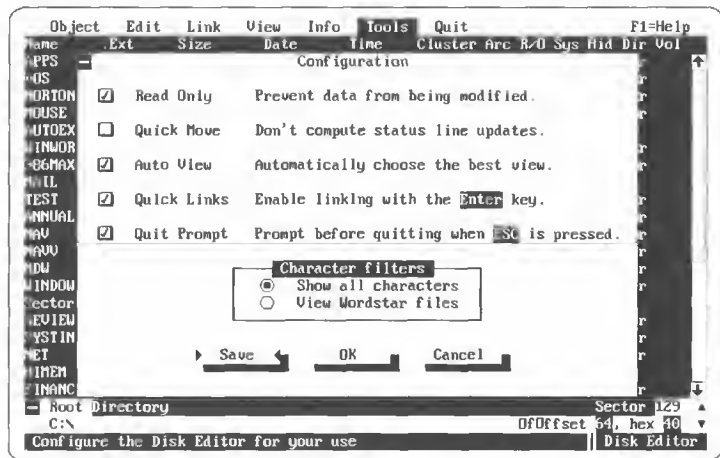
`DISKEDIT /?`

To get help while using the program press F1.

Configuration

Disk Editor configuration is controlled by background parameters that determine how the program behaves while you view and edit objects. You can control five of the items in the Disk Editor's CONFIGURATION... dialog box. To access this dialog box, pop up the Tools menu and select CONFIGURATION.... The default settings for the five items are shown in the figure below. To change any of the settings, click on it or move the cursor to it and press Spacebar.

Figure 2-1



Configuration dialog box

Read Only

Controls whether or not you can make changes to the selected disk. When it is checked, you can only view the object; you can't change it. This is the default setting. If you try to change anything, nothing happens. Also, the entire Edit menu is disabled, except MARK; in read-only mode you are allowed to mark (select) blocks of information and copy them to the clipboard (where they can be edited or saved to another disk).

Quick Move

Controls whether or not Disk Editor reports filenames on the status line as you move the cursor around in the object. By default, Quick Move is off so that Disk Editor does display filenames on the status bar. However, this takes time and may slow Disk Editor down. If you want to move faster, and don't need to see the name of the file you are in, try turning Quick

Move on. This is especially effective in the FAT viewer, where it suppresses not only the filename on the status line, but also the highlighting of the current cluster chain.

Auto View

Controls how Disk Editor selects a viewer for you. When it is on, which is the default, Disk Editor tries to select the most appropriate viewer for your object. For example, if you select a directory as the object, Disk Editor automatically brings up the directory viewer. If you select a file and Disk Editor determines that it is a text file, then it selects the text viewer. The hex viewer is selected when no other viewer is deemed more appropriate. When Auto View is off, the hex viewer is always selected.

No matter what viewer Disk Editor selects for you, you can always switch to any other viewer. The Auto View option affects only that initial decision by Disk Editor.

Quick Links

Disk Editor includes a link feature, which allows you to switch from one object to a related object (for example, from a FAT entry to the file itself) without going through the complete process of selecting a new object. The Quick Links option controls whether or not you can make this link simply by pressing Enter or double-clicking the mouse. By default, it is on, which means that whenever you press Enter or double-click the mouse, you link to a related object. Some people prefer not to link quite so easily, as they often press Enter by accident. If this is your preference, simply turn Quick Links off.

When you make unsaved changes to an object and then execute the link to a related object, you are prompted to Save, Discard, or Review the changes.

Controls how Disk Editor interprets character data. By default (Show All Characters), it uses all eight bits of a byte when determining the character to display; when the high-order bit (the eighth bit) is on, a graphics character results (some nondisplayable 8-bit characters will appear as dots). But WordStar and some related products use high-order bits for format information, not for graphics characters.

If you display a WordStar file using the default character translation, some characters will display incorrectly. Selecting the View Wordstar files option tells Disk Editor to ignore the high-order bits, which cleans up the display of WordStar files. However, it also prevents graphics characters from being interpreted correctly in other types of files.

If you use WordStar, you might want to set this option to View Wordstar files when displaying WordStar files but reset it to Show all characters when displaying any other types of objects.

Quit Prompt Character Filters

You now have the option to turn the prompt on or off when ESC is pressed to exit DiskEdit.

Save

Saves the current settings of the configuration options in the NU.INI file, making them permanent—at least until you change them again. The next time you start up Disk Editor, the same settings will be in effect. If you're running from a floppy, you may want to choose OK instead of Save.

OK

Puts the current settings into effect without saving them. They will last only until you exit Disk Editor.

Cancel

Closes the dialog box without making any changes to the configuration options.

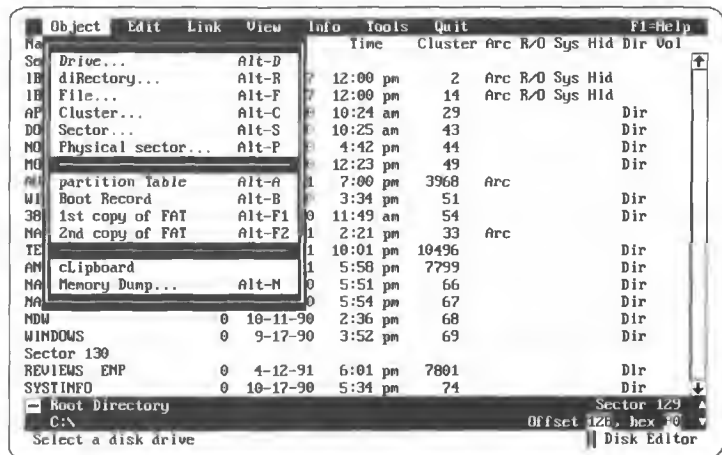
Selecting an Object

The Disk Editor can find and work with a given piece of disk data in a variety of ways, depending on its context and intended use. You could look at a piece of data, for example, by examining the file it is in, the disk cluster that contains it, or the sector that contains it. In addition to specifying the sector via DOS's logical sector-numbering scheme, you can also specify a physical sector number. (If the distinction between logical and physical sectors is new to you, you may read about these concepts in the "Disk Companion" chapter.) The primary difference between these various objects is in the way the data is addressed, or accessed: To find a file, you specify its name; to access a cluster, you indicate a cluster number or range; and to find a sector, you specify a sector number or range.

In general, your choice of object depends on what you know about the data in question and what you want to do with it. For example, if you want to check a file for damage, you would naturally select FILE... as the object; that way, all you would need to know is the file's name, not where its data is stored on the disk or in what order the clusters are strung together. On the other hand, if Norton Disk Doctor had just reported a bad cluster, you might want to use Disk Editor to examine that cluster; you would then select CLUSTER... as the object.

The first thing you should do after starting is select the type of object you wish to explore. You do this from the Object menu.

Figure 2-2



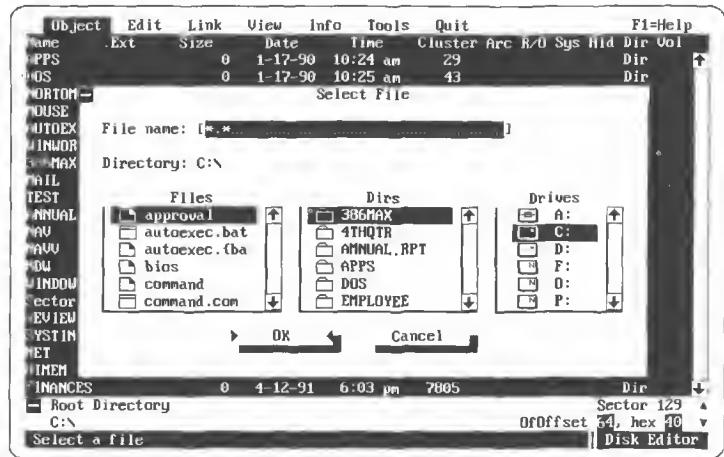
Object menu

Keyboard shortcuts exist for all the object options (except CLIPBOARD, which is discussed later). All the shortcuts on this menu are accomplished by pressing Alt with another key. The shortcuts are documented on the menu itself.

Selecting a File

To select a file, first select the FILE... option. The resulting dialog box shows the current drive, directory, and file list.

Figure 2-3



Select File dialog box

To select a different drive, directory, or file, highlight its name and press Enter or double-click on its name.

You can also select a file by typing in the text box at the top of the dialog box. You can enter a filename with or without wildcards, or as much of a path as you wish, as in C:\WORKERS\CLAIMS.DBF, or \SOURCE\WO*.C.

Identifying the Disk

You can select the DRIVE... option to change the current drive. You must do this before selecting any object other than a file that's on a different drive. You may do it for files also, although you can change drives in the file dialog box.

When you select DRIVE..., the resulting dialog box lists all the drive names; all you have to do is select the one you want.

The PHYSICAL DISK... option lets you access the disk without regard to its formatting or partitioning. Use it to edit disks that cannot be accessed by DOS, either because they were not formatted by DOS or because the system areas are damaged. When you select PHYSICAL DISK..., the list of drives will change into a list of disks. The only objects available to you will be physical sectors.

The **PHYSICAL DISK...** option also lets you access inactive areas on hard disks (areas not in any partition), and it lets you access an entire hard disk that has been partitioned into two or more drives.

Suppose you want to select the entire disk as the object. Selecting the drive from this dialog box is not enough; it identifies the drive or physical disk holding the object, but does not select an object. To examine and edit the entire disk, specify all the sectors as the object.

Selecting a Directory

To select a directory as an object, select the **DIRECTORY...** option from the Object menu. The resulting dialog box shows you the entire directory tree for the current drive. All you have to do is select the directory you want.

The dialog box contains a Speed Search box that you can use to search for the desired directory when it isn't shown on the screen. You don't have to place the cursor in the box. Simply type the first few characters of the desired directory name; the characters you type will appear in the Speed Search box, and the highlight will jump to the next directory in the list starting with those characters. The more characters you type, the more you pinpoint the directory you want. Disk Editor refuses to type any character that doesn't match a directory name; if you try to type a character and Disk Editor doesn't accept it, it means that there is no directory beginning with the characters you are trying to specify.

If the highlight moves to a directory that you don't want, you can press **Ctrl-Enter** to see the next directory in the list that starts with the Speed Search characters. Be sure to use **Ctrl** with the **Enter**; if you just press **Enter**, you select the highlighted directory as the object.

Selecting a System Area

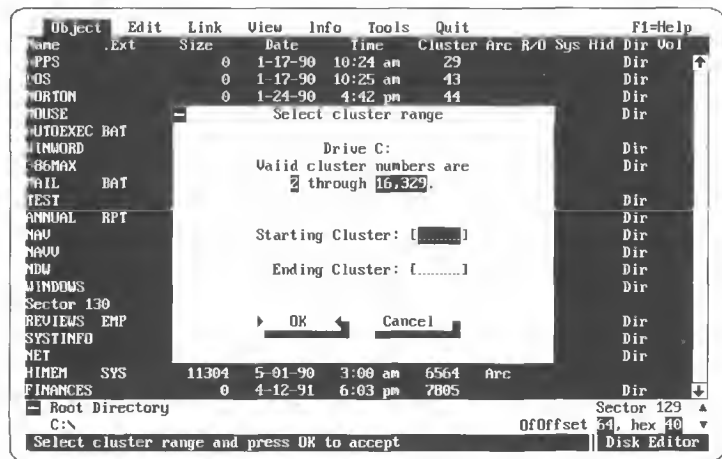
The system areas of a disk contain the Partition Table, Boot Record, and two copies of the FAT (see the "Disk Companion" chapter for an explanation of these terms. These areas are probably the most common areas to edit, so it would be a good idea to make sure that you understand what they contain.) The Object menu contains the commands to select the Partition Table, Boot Record, 1st Copy of FAT, and

2nd Copy of FAT. To edit any of these system areas, select the appropriate option from the menu or use the keyboard shortcut.

Selecting Clusters

You can identify any clusters on the disk as an object by selecting CLUSTER... on the Object menu. The resulting dialog box shows the valid cluster numbers and asks for a starting and ending number. There are no defaults.

Figure 2-4



Select Cluster Range dialog box

To view a group of consecutive clusters, enter the first number in the Starting Cluster box and the last number in the Ending Cluster box. To view a single cluster place its number in both boxes. If you include a starting cluster number but no ending cluster number, all the clusters on the disk are selected as the object; the first cluster you see is the one you specified, but you can scroll from there to any cluster on the disk.



TIP: Don't select more clusters than you need. If you need to make a backup copy of the object onto a diskette before you edit it, a smaller object is easier and faster to handle.

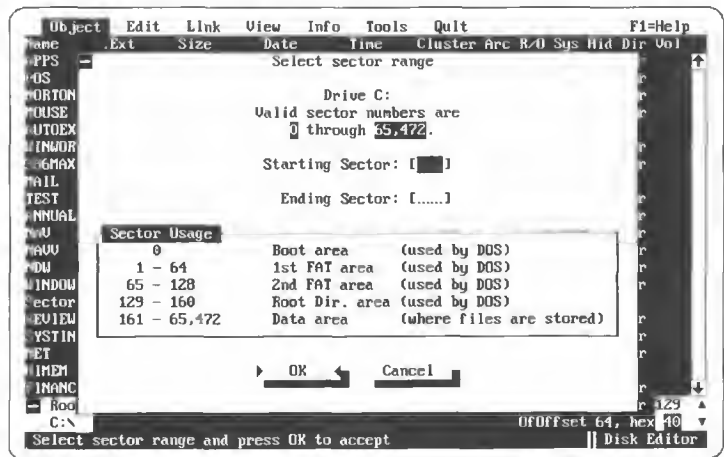


NOTE: Be sure to use Tab to move between the Starting Cluster box and the Ending Cluster box. If you press Enter after typing a starting cluster number, you will complete the dialog box with only a starting number, thus selecting the entire range of clusters. If you've selected the Ctrl-Enter option this will work differently, because Ctrl-Enter changes the way you exit from a dialog. Normally, the Tab key is used to move between fields and Enter accepts the dialog. When the Ctrl-Enter option is selected, Enter moves to the next field (like Tab), and you must press Ctrl-Enter to accept the dialog.

Selecting Logical Sectors as Objects

Selecting SECTOR... from the Object menu results in a dialog box where you can specify sector numbers. The box shows the range of sector numbers and lets you enter a starting and ending sector number. There are no defaults. This box works just like the cluster dialog box. Notice that you can enter sector numbers starting at 0, while cluster numbers start at 2.

Figure 2-5



Select Sector Range dialog box

Selecting Physical Sectors as Objects

Some programs report problems by physical sector coordinates rather than logical sector number. If you select PHYSICAL SECTOR... on the Object menu, the resulting dialog box lets you enter the cylinder, side, and sector number for the starting sector, and the number of sectors. Legal values for each item are displayed.

When the /M switch is specified on the command line, Disk Editor will attempt to access the diagnostic cylinder of any physical hard disk. If one is present, Disk Editor will allow you to access that cylinder.

Related Functions— The Diagnostic Cylinder

The Norton Calibrate program uses the diagnostic cylinder on a hard disk to store disk data while performing pattern tests on a hard disk. This will destroy information on the diagnostic cylinder. There are a few programs that rely on data stored there, and they may fail. Disk Editor will therefore allow you to edit this cylinder to replace any important information.

You can read the diagnostic cylinder only if Disk Editor is in maintenance mode. The diagnostic cylinder is yet another area of the disk where data can be stored, as mentioned in the paragraph above. This diagnostic cylinder is usually used to test specific aspects of your computer. However, on some computers it does not exist. Diagnostic software will often use it for this purpose. Also, many PARK programs will move to this cylinder when they are asked to park. Additionally, many calibrate programs (like CALIBRATE and SPinRite) use this cylinder to hold important information, from one session to the next. So, if the program is interrupted, that is, if the machine is turned off or loses power while the program is running), when you come back to your computer, the program leaves off exactly where it was before. This “state” information is saved on that diagnostic cylinder.

The two important points here are:

The ability to read the diagnostic cylinder means that you now have complete access to every part of your hard disk, whether it is used by DOS or not.

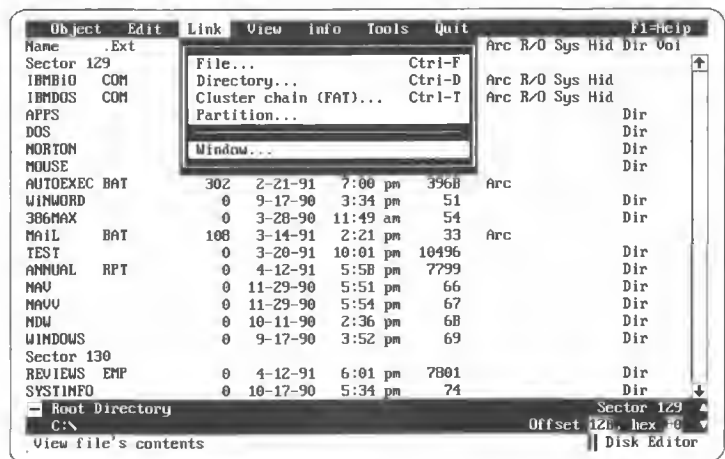
You now can see if any “special” data is stored on your diagnostic cylinder.

Linking to Related Objects

Files, their directory entries, and their FAT entries are closely related. When you are viewing or editing any one of these objects, you might need to examine the other two. Similarly, when you are trying to fix the Partition Table, you might need to access to the Boot Record and any other partition tables on the disk. Disk Editor lets you link (jump) directly to a related object without going through the Object menu and its dialog boxes.

Suppose you are examining the data in a file named CLAIMS.DB. It doesn't matter whether the current object is the file, the sector, or the cluster; Disk Editor knows that the cursor is currently in an area belonging to CLAIMS.DB. If you pull down the Link menu, two options will be available: DIRECTORY... and CLUSTER CHAIN (FAT).... Selecting DIRECTORY... immediately switches objects, taking you to the appropriate directory (and the directory viewer, if Auto View is on) with CLAIMS.DB highlighted. Selecting CLUSTER CHAIN (FAT).. takes you to the appropriate portion of the FAT, with the cursor on the cluster you were examining. (The cluster chain will be highlighted if Quick Move is off.) In either case, you can continue to examine the directory or the FAT, moving on to other areas that don't belong to CLAIMS.DB.

Figure 2-6



Link menu

Suppose you are examining a subdirectory named WORKERS using the directory viewer. The cursor is currently resting on the file named CLAIMS.DB. You can link from there to the file itself or to the FAT. If you link to the FAT, the cursor will be on the first cluster belonging to CLAIMS.DB. If the cursor is on an unused or invalid directory entry, links won't work.

Similarly, if you are editing the FAT and the cursor is currently on a cluster belonging to CLAIMS.DB in the subdirectory WORKERS, you can link to either the file or the directory. If the cursor is on an unused or bad cluster, you can link to the cluster but not to a directory. If the cursor is on a cluster belonging to a directory instead of a file, you can link to the directory.

When you are editing the Partition Table, the Link menu contains only one available option: PARTITION.... Selecting it links to the partition table for the highlighted partition, if there is one. Otherwise, it links to the Boot Record for the highlighted partition. The converse is not true; you can't link from the Boot Record back to the Partition Table. To get back to the Partition Table, or anywhere else, you have to go through the Object menu.

If Auto View is on, then the appropriate viewer will be invoked when you link to a related object. Otherwise, the linked object will appear in the same viewer that you were using before the link. You can change viewers as needed.

When Quick Links is on, you may not need to use the Link menu. Just press Enter or double-click on the desired entry to make the link. Files and directories automatically link to each other, the FAT links automatically to the highlighted file, and the Partition Table links automatically to the highlighted Partition Table or Boot Record. With Quick Links on, you can examine the files in a directory very easily. Double-click on the first entry; examine the data; double-click on any location to return to the directory; double-click on the second entry; and so on. If you don't have a mouse, highlight the first entry and press Enter to examine the data; press Enter to return to the directory; move the highlight to the second entry and press Enter; and so on.

When you change objects by linking, you change sectors. If you have made editing changes to the current sector and have not saved them, you will see a dialog box that asks if you want to save, discard, or review the changes before the link is completed.

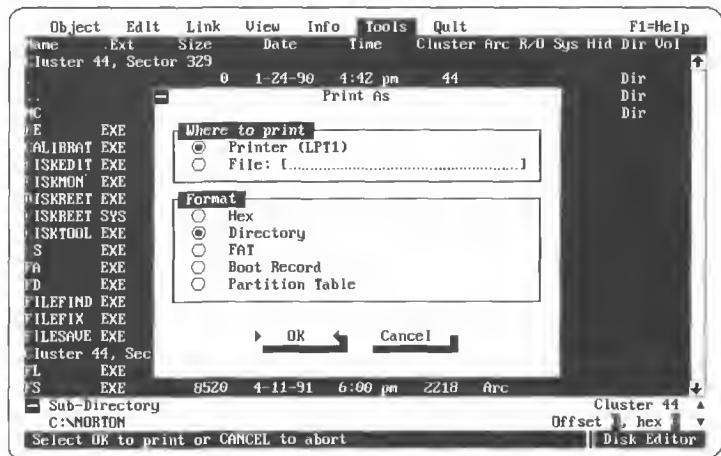
Print As ...

You can use Disk Editor to print any selection. From the Tools menu, select PRINT AS... to print.

The selection can be printed in either Hex, Directory, FAT, Boot Sector, or Partition Table format. The default print format will be the same as the current view, but the user can select a format via a set of radio buttons.

It's important to note that you can append printouts to each other by means of the overwrite/append dialog box.

Figure 2-7



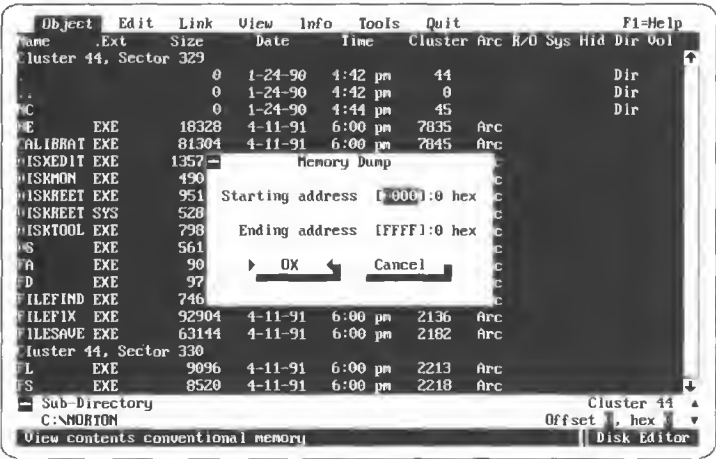
Print As dialog box

You can write data directly to your printer or to a file. Make your selection by highlighting one of the radio buttons. Choose the format in which you'd like your selection to be printed by highlighting the appropriate radio button. Click OK when you have made your selection(s). Choose Cancel to stop.

Memory Dump

You have the ability to view, search, and copy data from conventional memory. Working from a dialog box you will be prompted for a starting address and an ending address.

Figure 2-8



Memory Dump dialog box

Addressable memory is the first megabyte of memory. Ranges are specified in 16-byte increments.

There are two important uses of this feature: 1) browse RAM, and 2) recover data from RAM. The second option is especially important. If your application crashes and you end up back at the DOS prompt, for example, there is still data in RAM which you can salvage by browsing memory and writing chunks to disk files.

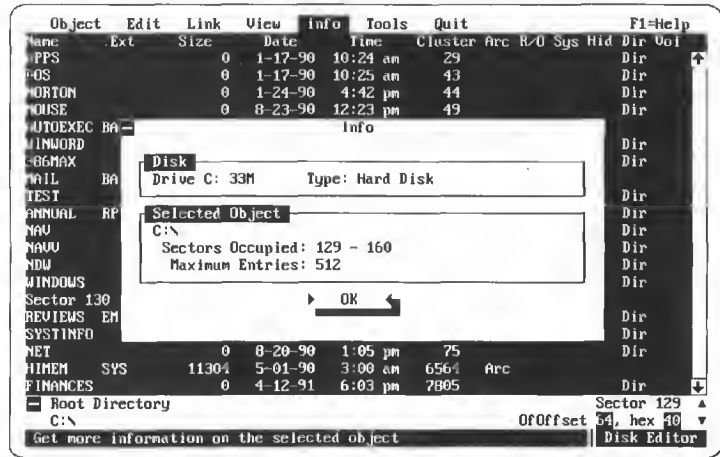
STOP

WARNING: It's possible that viewing certain areas of memory can cause the machine to lock up. This is especially true for IBM PS/2 computers. Because this is such a sensitive area in which to work, it shouldn't be touched unless you know what you're doing.

Displaying Information

Before you begin to actually edit the object, you should know some details about its structure and organization. The Info menu gives you access to information about the current object and the drive it's on.

Figure 2-9



Info menu

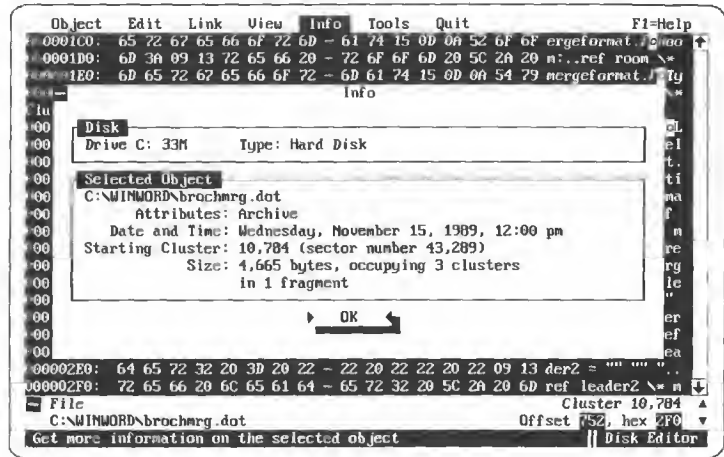
Object Information

To view information about the object itself, select the OBJECT INFO... option. The information presented varies depending on the type of object:

For a file or directory, the name, size, and type of drive are shown, as well as the filename or directory name, attributes, date and time of last modification, starting cluster and sector numbers, size, and number of clusters and fragments. In the case of network files or directories, cluster, sector, and fragment information is not displayed.

For a cluster, the drive name, size, and type are shown, as well as the cluster number, filename if appropriate, and the physical sector number.

Figure 2-10



Object Info screen for a file object

For a sector, the drive name, size, and type are shown, as well as the sector number, the filename if appropriate, and the physical sector number.

For a physical sector, the drive name, size, and type are shown, as well as the number of sectors selected.

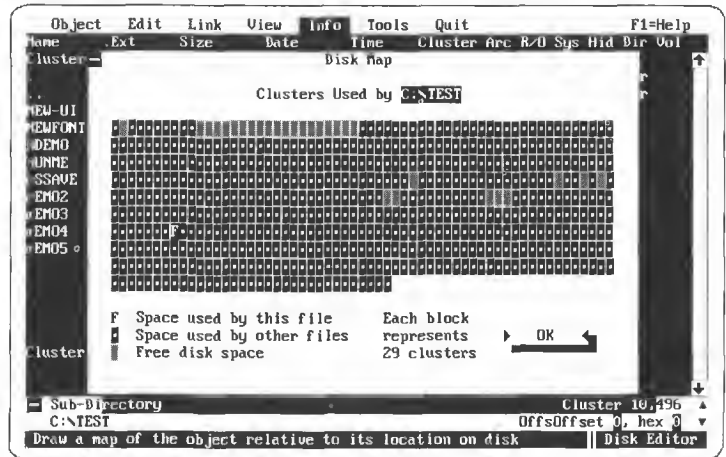
Drive Information

To display a technical report on the major characteristics of the current disk drive, select the **DRIVE INFO...** option. The report includes the drive name, size, type, logical characteristics (sector size, number of sectors per cluster, and so on), and the physical characteristics (sides, tracks, and so on).

Map of Object

To display a map of disk usage, select the **MAP OF OBJECT...** option. The map shows the entire range of clusters in the file area of the disk; the clusters occupied by the object are highlighted. You can get a rough idea of file fragmentation, free space, and the like, from the map. **MAP OF OBJECT...** is not available when sectors are selected.

Figure 2-11



Disk Map of fragmented file

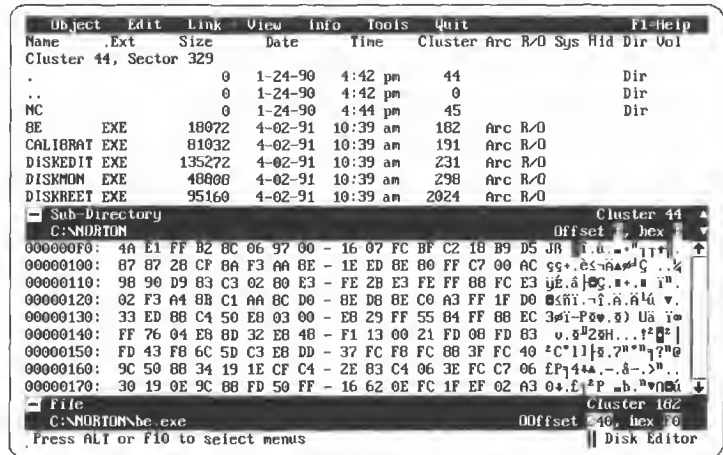
Editing Objects

There are five editable views: hex, directory, FAT, Partition Table, and Boot Record. The sixth view, text view, is not editable; if you wish to edit straight text, switch to hex format and edit in the text portion of the display. The main purpose of the viewers is to intelligently format the data in a way that makes working with it as easy as possible. If you were to view a directory object with the hex viewer, for example, the data structure would not be nearly as clear as it is when you view directory data with the directory viewer, as you can see below.

Selecting a View

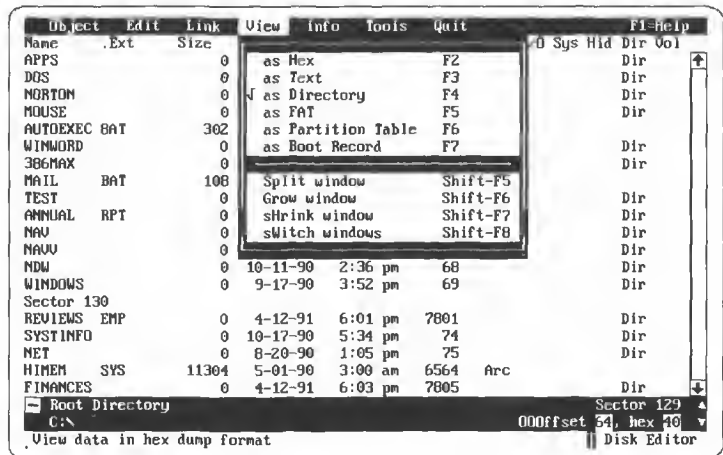
The viewer that Disk Editor initially uses to display the data depends on both the object and the status of Auto View. However, you are not locked in to that viewer. You can use the View menu to select another viewer. Disk Editor will show the data in the view you select, even if it makes no sense. It's up to you to make sure the viewer you select is appropriate for the data you are viewing/editing.

Figure 2-12



Directory view (top) and hex view (bottom) of a directory

Figure 2-13



View menu

All the items on the View menu have keyboard shortcuts. For the six editors, the shortcuts are single function keys. This makes it easy to switch views by simply pressing the appropriate function key.



TIP: You can sometimes identify what type of object you are looking at by trying out all the views until you find one where the data makes sense. Due to a disk write problem, for example, directory information may end up in a file, or vice versa. Also, the slack space of some files may contain pieces of other files.

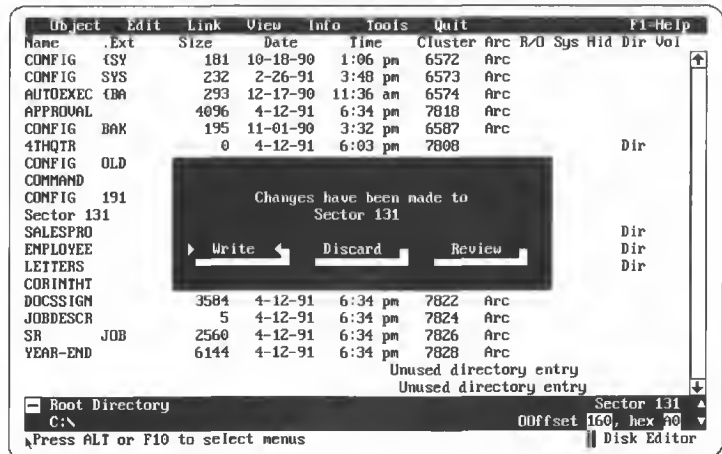
Making Changes

You can only make changes by overtyping existing data. You can never insert data. Each change is highlighted on the screen and remains highlighted until you save it, undo it, or discard it.

The Read Only configuration option must be off in order to make changes. See above for an explanation of this option.

Any time that you cross a sector boundary while editing—by scrolling to the next sector, selecting another object, or linking to a related object, for example—a dialog box opens up with the message “Changes have been made to sector/cluster [number].”

Figure 2-14

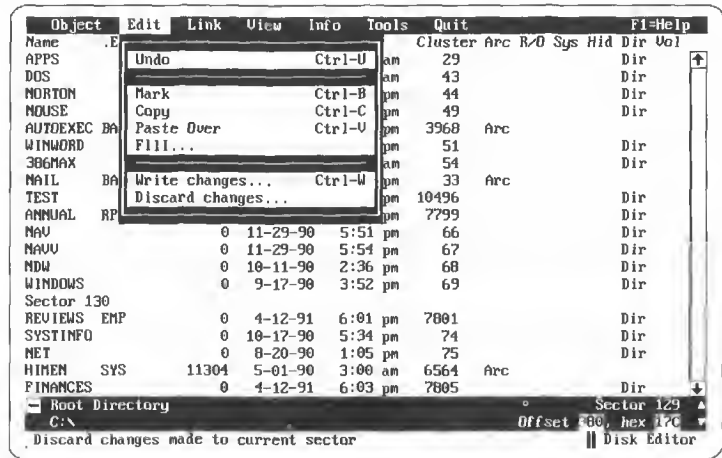


Write changes dialog box

You have the option of saving the changes that you have made so far. If you select Write, the sector you are leaving will be updated on disk. If you select Discard, you will lose the changes in the sector you are leaving, because Disk Editor cannot remember changes across a sector boundary. If you choose Review you remain in the current sector without saving the changes.

You don't have to cross a sector boundary to discard changes. You can select DISCARD CHANGES... from the Edit menu.

Figure 2-15



Edit menu

This gets rid of all unsaved changes in one fell swoop. In fact, it makes a new copy of the object in memory from the disk. Then you can start your editing all over again. However, you cannot discard changes that have already been saved on the disk.

Undoing Changes

Disk Editor includes an UNDO feature (in the Edit menu) that can remember changes for up to 512 bytes, as long as you don't cross a sector boundary. Changes are remembered in sequential order and undone in reverse order. You cannot undo changes that have been written to a disk.

Think of the UNDO feature like a stack of plates. Every time you change a byte, another plate is added to the top of the stack. When you select UNDO, the top plate is removed from the stack. Selecting UNDO again removes the next plate from the stack, and so on until the stack is empty. The stack can hold a total of 512 plates. If you add a 513th plate, the stack receives it, but the bottom plate drops off and is lost. If you cross a sector boundary, you have to start a new stack of plates.

To select UNDO, choose the Edit menu and select UNDO, or simply press Ctrl-U. Suppose you change four bytes in a sector, then realize that you changed the wrong bytes. Press Ctrl-U four times to restore the original values.

Saving Changes

You also don't need to cross a sector boundary to save changes. You can write changes to the disk, or you can save the entire object to another location.

Rewriting the Sector: To save your changes to the disk without leaving the sector, select WRITE CHANGES... from the Edit menu. This causes the contents of the edit buffer to be rewritten from memory to the disk, overwriting the former contents of the object.

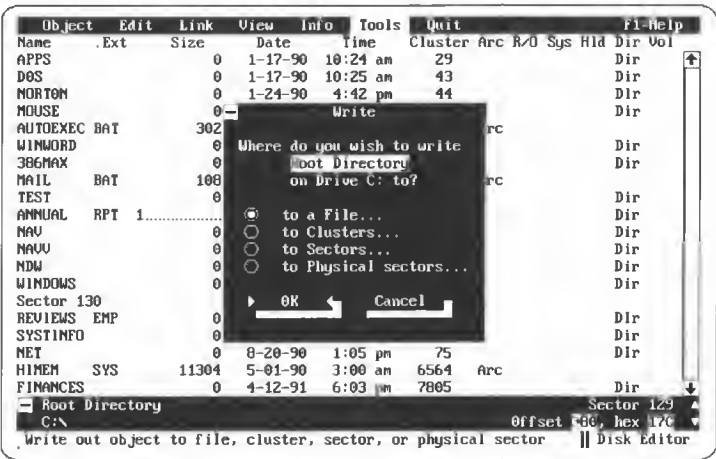
Creating a New Object

The WRITE CHANGES... command we just described rewrites the contents of the edit buffer back to the same location they came from on the disk. Sometimes, however, you need to write the object to a different location, perhaps even to a different disk. If, for example, you are creating a file out of some text you found in the unallocated space, you could not write the file with WRITE CHANGES.... Instead, you need the WRITE TO... option on the Tools menu.

You can also use this option to make a backup copy of an object before editing it. For example, suppose you plan to edit the Boot Sector. Select it as an object and save it to a diskette as a file named SAVEBOOT (or whatever name you prefer). Then go ahead and edit the Boot Record. Try out your new version; if it doesn't work, you can restore the former version from the file named SAVEBOOT.

The WRITE TO... dialog box lets you write the object as a file or specify the location by cluster, sector, or physical sector number. After you make your choice, Disk Editor asks for the name of the drive to write on. Then it asks for the filename or the starting cluster, sector, or physical sector number. For sectors, it shows you a table of sector usage on the target drive so you can see where the system areas and the file space are. Finally, it writes the object to the specified location.

Figure 2-16



Write to... dialog box

STOP

WARNING: You should know exactly what you are doing when using this feature. If you write directly to clusters or sectors, you might overlay valuable information that can never be recovered.

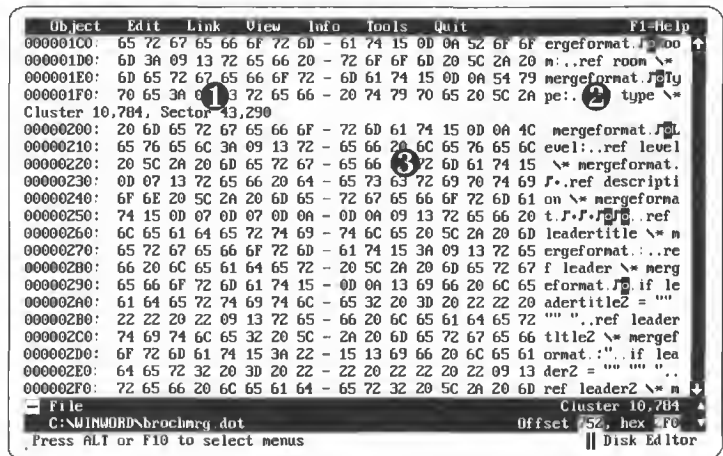
The Hex Viewer

The hex view is the main format for editing. It will appear when no other viewer is more appropriate or when Auto View is off.

Screen Layout

The main area is divided into three sections. The address shows the offset of the first byte on each line from the beginning of the sector. The hexadecimal section displays each byte as a two-character hexadecimal value. Sixteen bytes are displayed on a line, divided in the middle by a dash. The text section displays the same 16 bytes as text characters whenever possible. Some nondisplayable characters are shown as dots. Cluster or sector boundaries are clearly identified.

Figure 2-17



Hex view

1. hexadecimal display
2. text display
3. sector boundary

The status bar shows information about the object (on the left) and information about the current cursor position (on the right).

Navigating with the Mouse

Click on any position where you want to place the cursor. Use the scroll bar for scrolling. Left mouse button clicks in the shaded area above or below the scroll box “elevator” move one screenful at a time. Right mouse button clicks in the shaded area above or below the scroll box “elevator” move one sector at a time.

Navigating with the Keyboard

The cursor keys move the cursor around in the object, with the window scrolling as necessary. PgUp and PgDn move one screenful at a time. Ctrl-PgUp moves back one sector or cluster and Ctrl-PgDn moves forward one sector or cluster. Home jumps to the beginning of the object and End jumps to the end. You cannot scroll beyond the beginning or end of the selected object. (But you can select a bigger object.)

Making Changes

The hex and text sections each contain a cursor; the active cursor blinks while the inactive one is solid. If the active cursor is in the hex section, you can replace data by entering hex values. If the active cursor is in the text section, you can type any keyboard character. Any ASCII character can be entered by holding down the Alt key while you type the decimal code for the character on the numeric keypad. You can consult the ASCII table in the Tools menu to find the correct decimal codes.

The changes you make overlay existing data and are reflected in both sections. You can move the active cursor from one section to the other by pressing Tab. You cannot edit the addresses.

Undoing Changes

In addition to the standard UNDO feature, you can undo any change by backspacing over the changed data. In the hex window, you can undo one digit (one-half of a byte) by backspacing, whereas the UNDO feature undoes changes to a complete byte (one character).

The Directory Viewer

Use the directory viewer to view and edit a directory object.

Figure 2-18

Object	Edit	Link	View	Info	Tools	Quit	F1-Help				
Name	.Ext	Size	Date	Time	Cluster	Arc	R/O	Sys	Hid	Dir	Vol
APPS		0	1-17-90	10:24 am	29					Dir	
DOS		0	1-17-90	10:25 am	43					Dir	
NORTON		0	1-24-90	4:42 pm	44					Dir	
MOUSE		0	8-23-90	12:23 pm	49					Dir	
AUTOEXEC	BAT	302	2-21-91	7:00 pm	3968	Arc					
WINWORD		0	9-17-90	3:34 pm	51					Dir	
386MAX		0	3-28-90	11:49 am	54					Dir	
MAIL	BAT	108	3-14-91	2:21 pm	33	Arc					
TEST		0	3-20-91	10:01 pm	10496					Dir	
ANNUAL	RPT	0	4-12-91	5:58 pm	7799					Dir	
MAU		0	11-29-90	5:51 pm	66					Dir	
MAUV		0	11-29-90	5:54 pm	67					Dir	
MDW		0	10-11-90	2:36 pm	68					Dir	
WINDOWS		0	9-17-90	3:52 pm	69					Dir	
Sector 130											
REVIEWS	EMP	0	4-12-91	6:01 pm	7801					Dir	
SYSTINFO		0	10-17-90	5:34 pm	74					Dir	
NET		0	8-20-90	1:05 pm	75					Dir	
HINEN	SYS	11304	5-01-90	3:00 am	6564	Arc					
FINANCES		0	4-12-91	6:03 pm	7805					Dir	

root Directory Sector 130
C:\ Offset 12E, hex 86
Press ALT or F10 to select menus Disk Editor

Directory view

Screen Layout

Each line shows one directory entry, with the fields separated and labeled. Dates and times are formatted. Attributes are displayed as follows:

Arc	Archive
R/O	Read only
Sys	System
Hid	Hidden
Dir	Directory (the entry is a subdirectory, not a file)
Vol	Volume label

Unused entries, identified by hex 00 in the first byte, are labeled Unused directory entry. Erased entries have the hexadecimal number E5 (lowercase Greek sigma) as the first character. Cluster or sector boundaries are clearly marked.

Entries marked as "Invalid directory entries" have invalid values in some of the fields; for example, they have invalid characters in the filename. Usually, invalid directory entries appear when the directory view is used on a non-directory object, such as a boot record or file.

Navigating with the Mouse

To navigate with the mouse, simply click where you want to place the cursor. You can scroll using the scroll bar.

Navigating with the keyboard: When Read Only mode is off, move between fields by pressing Tab and Shift-Tab. Move between lines by pressing Up Arrow and Down Arrow. Move within a field by pressing Left Arrow and Right Arrow. Home jumps to the first entry, and End jumps to the last. PgUp and PgDn move one screenful at a time. On the numeric keypad, - and + move one sector or cluster at a time. When Read Only mode is on, Left Arrow and Right Arrow move from field to field, and not within fields.

Making Changes

To change a field, move the cursor to it and overwrite it. You can edit (insert and delete characters) to a certain extent, as long as you don't exceed the size of the field. For example, you can change a month from 1 to 10 by adding a 0 and you can change the name of a file from CLAIMS to RECLAIMS by inserting RE in front of the name.

The directory viewer will not let you enter an invalid value in any field. This is usually an advantage. However, if you need to bypass this feature for any reason (and you know what you are doing), you can switch to the hex viewer, which places no limitations on the values you enter in bytes, even though the object you are editing is a directory.

You do not need to type values into any of the attribute fields. Simply toggle an attribute on and off by positioning the highlight in the correct field and pressing Spacebar.

You can undo, discard, and write changes using the UNDO, WRITE CHANGES..., and DISCARD CHANGES... options on the Edit menu.

**STOP**

WARNING: Be careful when editing a directory; if you make a mistake, DOS may not be able to access the files correctly. Make a backup copy of the directory before you edit it, so that you can restore it if your editing doesn't have the effect you want.

Changing Attributes

You can change attributes for an individual entry by positioning the cursor and pressing the Spacebar or by clicking on the correct position. However, if you want to set the attributes for a group of adjacent entries, it's easier to use the SET ATTRIBUTES... option on the Tools menu. Highlight the group of entries as a block (see the section on the clipboard—under “Power Editing Features”—for instructions on marking blocks). Then select SET ATTRIBUTES... and indicate which attributes you want to set and which you want to clear in the resulting dialog box.

Setting the Date and Time

You can edit the date and time for an individual entry by overtyping the original values. But if you want to set the date and time for a group of adjacent entries, it's easier to do it with the SET DATE/TIME... option on the Tools menu. Highlight the group of entries that you want to affect. Then select SET DATE/TIME... and enter the desired date and time in the resulting dialog box.

The Boot Record Viewer

The Boot Record viewer formats a boot record and labels the fields so that it is easy for you to interpret and edit its data. It even suggests values for the various fields to help you determine whether or not you are looking at a legitimate Boot Record.

Screen Layout

The Boot Record view shows each field in the boot record with the values from the current object. The DOS Reports column contains the values that DOS is currently using. Boot Record contents have grown with each version of DOS, so in earlier DOS versions the last fields are unused.

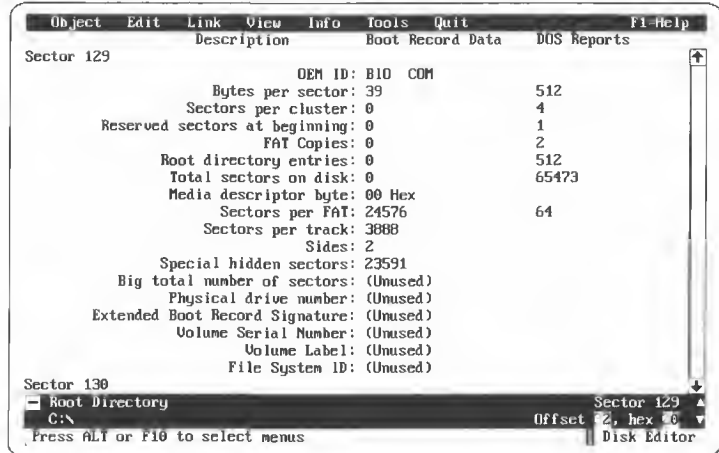
Navigating with the Mouse

To navigate with the mouse, simply click where you want to place the cursor. You can scroll using the scroll bar.

Navigating with the Keyboard

Move between fields by pressing PgUp and PgDn. Move within a field by pressing Left Arrow and Right Arrow, which work only when Read Only is off. Home jumps to the first entry, and End jumps to the last.

Figure 2-19



Boot Record view

Making Changes

You can change a value by overtyping it. You can edit a value to a certain extent; for example, you can change 5 to 50 by adding a 0. You can undo, discard, and write changes using the UNDO, DISCARD CHANGES..., and WRITE CHANGES... options of the Edit menu.

The viewer will prevent you from entering some illegal values, but it will not prevent illogical ones. You can bypass the restrictions if necessary (and if you know EXACTLY what you are doing) using the hex viewer, which places no restrictions on the values you enter.

STOP

WARNING: Be careful when changing any Boot Record, but especially the Boot Record of your hard disk. If you change any value, make sure you know what you are doing. You could damage your ability to boot from the changed disk or to access its files. It is a good idea to make a backup copy of the Boot Record you want to edit. Also, if you have installed the Image utility, it makes a copy of the Boot Record.

The FAT Viewer

The FAT viewer displays the File Allocation Table in a manner that is convenient to read and edit.

Figure 2-20

Object	Edit	Link	View	Info	Tools	Quit	F1=Help
Sector 129							
16969	16973	20297	8224	20291	10061	0	0
0	0	0	24576	3888	2	23591	0
16969	17485	21327	8224	20291	10061	0	0
0	0	0	24576	3888	14	30632	0
20545	21328	8224	8224	8224	4128	0	0
0	0	0	21262	5169	29	0	0
20292	8275	8224	8224	8224	4128	0	0
0	0	0	21307	5169	43	0	0
20302	21586	20047	8224	8224	4128	0	0
0	0	0	34135	5176	44	0	0
20301	21333	8261	8224	8224	4128	0	0
0	0	0	25322	5399	49	0	0
21825	20308	22597	17221	16706	8276	0	0
0	0	0	38922	5717	3968	302	0
18775	22350	21071	8260	8224	4128	0	0
0	0	0	31826	5425	51	0	0
14387	19766	22593	8224	8224	4128	0	0
0	0	0	24106	5244	54	0	0
16717	19529	8224	8224	16706	8276	0	0
0	0	0	29345	5742	33	108	0

Root Directory Sector 129
 Drive C: Cluster 3,493,449,747, hex D039D013
 Press ALT or F10 to select menus Disk Editor

FAT view

Screen Layout

Each field represents one FAT entry and contains one of the following:

- 0 Unused cluster
- 2-... Number of next cluster in chain (maximum varies from disk to disk)
- <EOF> Last cluster in chain ("end of file" marker)
- <BAD> Bad cluster

If Quick Move is off, the current cluster chain (the one containing the cursor) is highlighted. The status bar shows the FAT number and the name of the file containing the cluster on the left, the sector number and the current cluster number (in decimal and hex) on the right.

Navigating with the Mouse

To navigate with the mouse, simply click where you want to place the cursor. You can scroll using the scroll bar.

Navigating with the Keyboard

Move between clusters by pressing Tab and Shift-Tab. Move between lines by pressing Up Arrow and Down Arrow. Move within a field by pressing Left Arrow and Right Arrow, which work only when Read Only is off. Home jumps to the first entry, and End jumps to the last. PgUp and PgDn move one screenful at a time. Ctrl-PgUp and Ctrl-PgDn move one sector at a time.

Making Changes

You change a FAT value by overtyping it. You can also edit the existing value to a certain extent; for example, you can change 5 to 50 by adding a 0. You must take care when editing FAT entries not to create a cross-link, where the cluster chains of two different files include a common cluster. Type E to enter an <EOF> value and B to enter a <BAD> value.

You can undo, discard, and write changes using the UNDO, WRITE CHANGES..., and DISCARD CHANGES... options on the Edit menu.

STOP

WARNING: Don't edit the FAT unless you know what you are doing. If you make a mistake, you may make it impossible for DOS to access your files correctly—or at all. It's always a good idea to back up your disk before editing sensitive areas of the disk. Our Image program, when installed, makes a copy of the FAT. DOS also keeps a second copy of the FAT.

The Partition Table Viewer

Screen Layout

The Partition Table viewer displays the Partition Table data in a manner that is easy to interpret and edit.

Each line in the table describes one partition on the disk. The System column shows the operating system that owns the partition:

DOS-12	DOS (12-bit FAT)
DOS-16	DOS (16-bit FAT)
EXTEND	DOS (extended partition)
BIGDOS	Compaq DOS 3.31, DOS 4.0, and any disk larger than 32MB
XENIX	XENIX

BBT	UNIX (Bad Block Table)
386-ix	UNIX
PCIX	PCIX
HPFS	High Performance File System (OS/2)
Split	SplitDrive
Speed	SpeedStor
DM	Disk Manager
GB	GoldenBow VFeature
NET286	NOVELL
NET386	NOVELL
CP/M	CP/M
?	unrecognized operating system
Unused	unused partition entry

Figure 2-21

Object Edit Link View Info Tools Quit FI-Help										
System	Boot	Starting Location			Ending Location			Relative Sectors	Number of Sectors	
		Side	Cylinder	Sector	Side	Cylinder	Sector			
DM	No	0	80	0	83	32	32	538976288	4128	
unused	No	0	0	0	0	0	0	351836692	2671	
DM	No	0	80	0	83	32	32	538976288	4128	
unused	No	0	0	0	0	0	0	351836692	2671	

Root Directory	Sector 131
Drive C:	Offset 162, hex 1CE
Press ALT or F10 to select menus	
Disk Editor	

Partition Table view

The Boot column shows whether or not the partition is bootable. Only one entry should say yes.

The Starting Location and Ending Location columns show the physical coordinates of the first sector and last sector in the partition. The Relative Sectors column shows the starting location of the sector as a relative (DOS-style) sector number. The Number of Sectors column shows the total number of sectors in the partition.

***Navigating with
the Mouse***

To navigate with the mouse, simply click where you want to place the cursor. You can scroll the cursor up and down using the scroll bar.

***Navigating with
the Keyboard***

Move between fields by pressing Tab and Shift-Tab. Move between lines by pressing Up Arrow and Down Arrow. Move within a field by pressing Left Arrow and Right Arrow which work only when Read Only is off. Home jumps to the first entry, and End jumps to the last. PgUp and PgDn have no effect.

Making Changes

You can change any value by overtyping. You can edit values to a certain extent. You can toggle the Boot column from No to Yes by positioning the cursor on the current value and pressing Spacebar. The Partition Table viewer will not let you enter an illegal value into any field.

You can undo, discard, and write changes using the UNDO, WRITE CHANGES..., and DISCARD CHANGES... options on the Edit menu.

***Recalculating
the Partition***

If you enter the Starting Location and Ending Location, you can ask Disk Editor to calculate the Relative Sector and Number of Sectors by selecting RECALCULATE PARTITION from the Tools menu. You must zero the Relative Sectors and Number of Sectors fields before recalculating them; if you do not, a dialog box appears to tell you to do so.

**STOP**

WARNING: Careless editing of the Partition Table is potentially very dangerous. In general, the proper way to make changes to the Partition Table is via the DOS FDISK utility. The Disk Editor facility is provided to deal with cases that FDISK doesn't cover. For example, you can fix errors that have crept into the table and you can view and edit tables created by operating systems that FDISK doesn't recognize.

Do not make changes to the Partition Table unless you know exactly what you are doing. It's always a good idea to back up your disk before writing to sensitive areas such as the Partition Table. The Image utility, if installed, makes a copy of the Partition Table.

Power Editing Features

Disk Editor offers several features, available in all views, that make formerly complex editing jobs a snap. The clipboard lets you copy blocks of data, the fill feature overwrites existing data with a fill character, and the split window feature lets you work with two objects and editors at once.

The Clipboard

The clipboard is a 4096-byte memory buffer that lets you copy blocks of data from one place to another. It can be used with any viewer. To use it, mark a block of data, copy it to the clipboard, move the cursor to the target location, and paste the block from the clipboard.

You can paste data across a sector boundary; however, you will be warned that your changes will not be undoable. You can cancel the changes at that time, if you wish.

You can mark a block using a mouse by dragging the pointer over the desired block. The block will be highlighted as you drag the pointer. (In hex view, it will be highlighted in both the hex and ASCII sections.) With the keyboard, position the cursor at either end of the block, select MARK from the Edit menu, and move the cursor to the other end of the block. You will see the block highlight grow as you move the cursor.

You can remove the block highlight by clicking anywhere or by selecting MARK again, followed by any cursor key.

To copy the block to the clipboard, select COPY from the Edit menu. When the menu disappears, the block has been copied. It completely replaces the former contents of the clipboard, even if the clipboard was full before and the new block is only one byte. You cannot append data to the clipboard with the COPY command.

If the block is too large for the clipboard—if it's larger than 4096 bytes—you will see an error message. You must respond to the message, then adjust the block to be a smaller size. 4096 bytes is 256 lines in the hex viewer, or eight sectors (two clusters on most hard disks).

Data stays in the clipboard until you overwrite it with another block or terminate Disk Editor. Therefore, you don't have to paste the block immediately. You can switch objects, switch disks, examine Boot Records, and so on until you are ready to paste it.

To paste, position the cursor and select PASTE OVER from the Edit menu. The entire contents of the clipboard are written into the object starting at the cursor position, overwriting the former values. Pasting does not empty the clipboard. The block of data remains on the clipboard and can be pasted in as many other locations as needed.

Each time you paste, highlights show you which bytes were actually changed by the pasted block. UNDO undoes the entire pasted block, restoring all the former values, not just one byte.

The clipboard can be selected as an object when it contains data. Select CLIPBOARD from the Object menu, then continue to view and edit it like any other object. When you paste, the edited version will be pasted.

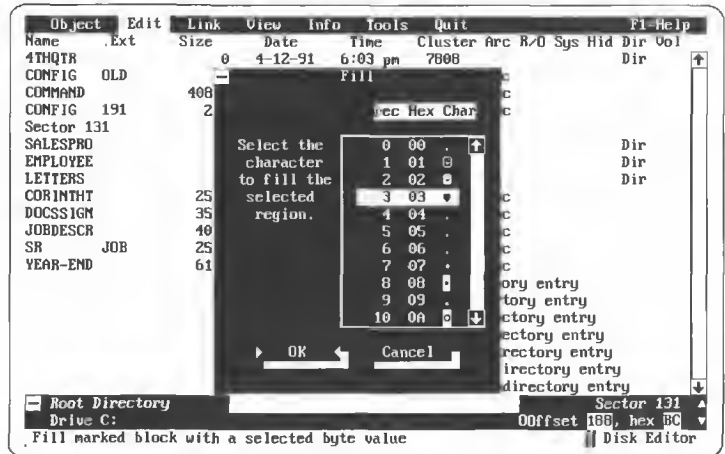
Filling Bytes with a Specified Character

You can fill a marked portion the current object with a specified character; the same character is placed in every byte in the block. MARK the entire area to be filled as a block. Then select FILL... from the Edit menu. A dialog box asks for the fill character, giving you a table of characters to choose from.

When you highlight the desired fill character and select OK, every byte in the highlighted block is overwritten with the specified character.

There is no limit on block size when filling; you can cross sector boundaries. If you do, however, you will be warned that your changes will not be undoable. You can cancel the changes at that time, if you wish.

Figure 2-22



Fill dialog box

Suppose you want to send a file to a colleague, but you don't want to share some sensitive information in the middle of the file. Copy the file to the diskette you will send and select the new copy (the diskette copy) as an object. Locate the confidential data and mark it as a block. Select **FILL...** and select a meaningless value (hex 00 or hex 20 would be good). When you select **OK**, any former contents of that area on the diskette are effectively eradicated. (You should consider converting the file you want to send to ASCII format, so that when you fill bytes in the middle of the file, you will not delete essential formatting information.)

Filling entries in a FAT table is slightly different than other fills, because two bytes are needed for each entry, instead of one byte. A special dialog box reminds you of this when you start to use the fill feature in a FAT.

Split Windows

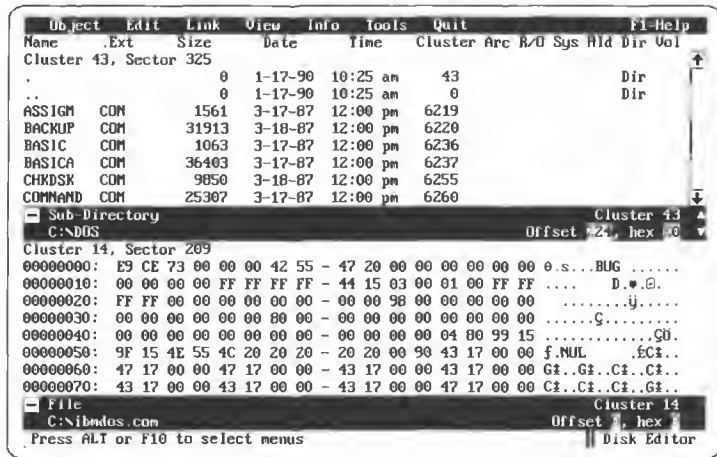
You can view two objects at once by opening a second window on the screen. You can switch the cursor back and forth to examine and edit both objects. You can even establish a dynamic link between the two windows so that changes in one are automatically reflected in the other. Disk Editor will also compare the two objects, highlighting any bytes that are different.

To open the second window, select **SPLIT WINDOW...** from the View menu. The second window will appear in the bottom half of the screen, containing the same object and view as the first window. The top window's status bar separates the two windows. You can change the contents of either window.

It's even easier to open a second window with the mouse. Simply drag the original status bar up the screen. Position the mouse pointer anywhere on the bar, press the mouse button, and move the pointer up while you hold the button. You will see the bar move and the second window open up.

When the second window appears, it contains the same object and view as the first window. Activate the window and select whatever object and view you need. The contents of the window and status box will change accordingly, without affecting the other window.

Figure 2-23



Viewing two objects in a split window

Only one window at a time can be active. The active window contains the cursor and is the focus of all keyboard and menu actions. You can identify the active window by the position of the cursor and by the presence of a scroll bar, if appropriate. To activate the other window, click on it or select **SWITCH WINDOWS** from the View menu.

You can adjust the sizes of the windows by dragging the top status bar. If you don't have a mouse, select either `GROW WINDOW` or `SHRINK WINDOW` from the View menu. These two options adjust the size of the active window by one line. The smallest window size is one line plus the status bar. You cannot shrink a window smaller than that using either the mouse or the `SHRINK WINDOW` option. Some views have even larger minimum sizes because of the need to display labels along with at least one line of data.

To close the active window, select `UNPLIT WINDOW...` from the View menu. With the mouse, you can close either window by clicking on the appropriate close box.

Comparing Two Windows

To compare the contents of two windows, position the cursor on the byte where you want the comparison to start. Then select `COMPARE WINDOW...` from the Tools menu. You will see a message about the two objects being compared, with a Stop option. Starting at the cursor position, Disk Editor compares each pair of bytes until it finds a mismatched pair. It positions the cursor on the unmatched byte in the active window and terminates the comparison. You can then examine the mismatched bytes and decide what to do. You can continue the comparison from that point forward by selecting `COMPARE WINDOWS...` again.

Suppose you want to see if the first and second FAT are identical. Load the first copy into the top window and the second copy into the bottom window. Put the cursor on the first cluster in either window and select `COMPARE WINDOWS...`. If the comparison stops with a cluster highlighted, the two copies don't match at that point.

Linking Windows

Certain types of objects can establish dynamic links to the other window. For example, if you have a FAT in the top window and select `WINDOW...` from the Link menu (so that a check mark appears next to `WINDOW...`), the bottom window will always show the contents of the cluster that is highlighted in the top window. As you move the highlight in the top window, the contents of the bottom window changes accordingly.

Similarly, if the top window contains a directory and you select the WINDOW... option on the Link menu, the bottom window will show the contents of the file highlighted in the top window. As you move the highlight, the file in the bottom window changes.

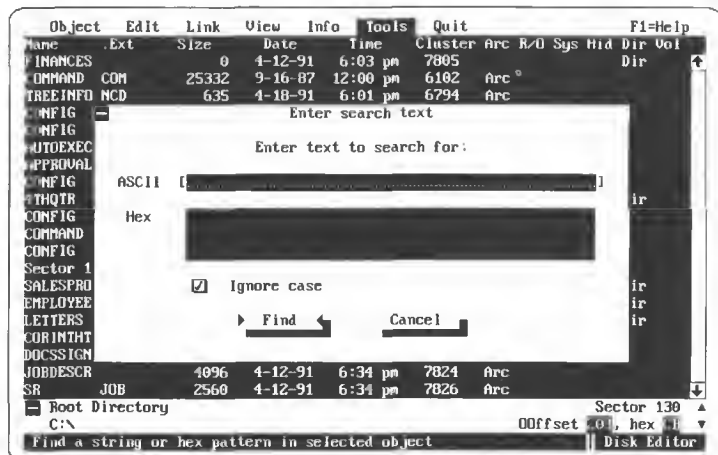
To break the dynamic link, select the Link menu's WINDOW... option again; the check mark will disappear.

Searching for Specific Data

You can search an object for a specific string of data. Select the FIND... option from the Tools menu. The resulting dialog box asks for the search string of up to 48 bytes. You can type it in hex or ASCII (or a combination of both). If you select Ignore case, then the difference between uppercase and lowercase letters is ignored; "Norton" matches "NORTON," for example. Turn Ignore case off for hex data; otherwise hex 41 (ASCII "A") will match hex 61 (ASCII "a").

When you select OK, Disk Editor searches forward from the current cursor position until a matching string is found. It highlights the string and terminates the search function. If the object is large and the desired string isn't found immediately, a dialog box appears that tracks the progress of the search through the object. You can discontinue the search by selecting Stop on this dialog box.

Figure 2-24



Find... dialog box

Disk Editor saves the search string until you enter another one or terminate Disk Editor. You can continue the search at any time by selecting FIND AGAIN, which searches forward from the current cursor position using the saved search string. The FIND AGAIN shortcut key, Ctrl-G, lets you quickly repeat a search until you find the instance you are looking for.

Tools

The Tools menu offers several facilities that make editing, and even exploring, easier.

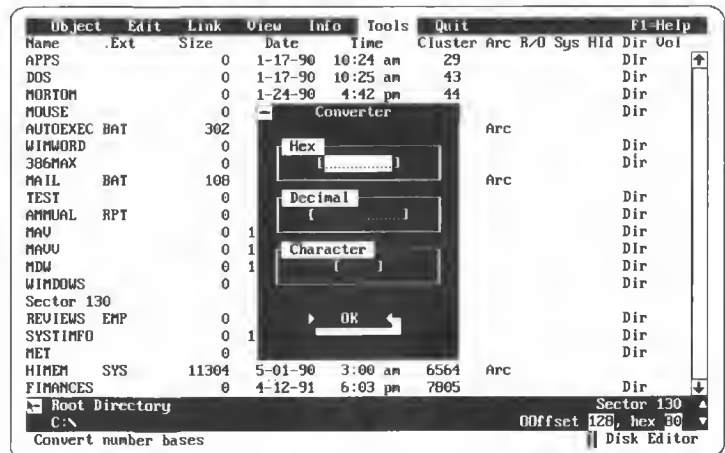
Hex Converter

The hex converter converts values between hexadecimal and decimal. The character (ASCII) equivalent of each hex byte is also shown. To use the converter, select HEX CONVERTER... on the Tools menu. In the dialog box, you can enter a value in any of the three formats. The value is instantly converted into the other two formats.

Suppose you're examining a byte in the Boot Record in hex view and the byte is A2. To find out the decimal equivalent of this number, open the hex converter and enter the number (the cursor is in place to accept hex input). The decimal field will show 162.

The hex converter can accept eight hex digits, ten decimal digits, or four ASCII characters.

Figure 2-25

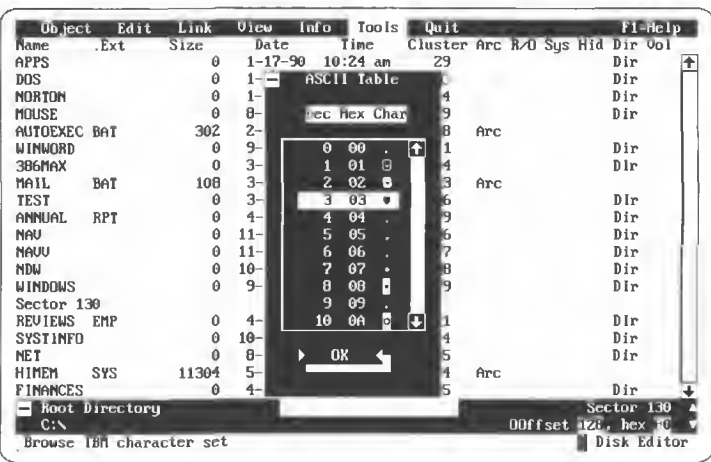


Hex converter... dialog box

ASCII Table

You can call up a table of ASCII characters and their hexadecimal and decimal equivalents by selecting the ASCII TABLE... option on the Tools menu. You can scroll through the table to find the character you want. The table is for information only; you can't process it or automatically copy characters from it.

Figure 2-26



ASCII table

Exiting the Disk Editor

You can exit Disk Editor through the options on the Quit menu.

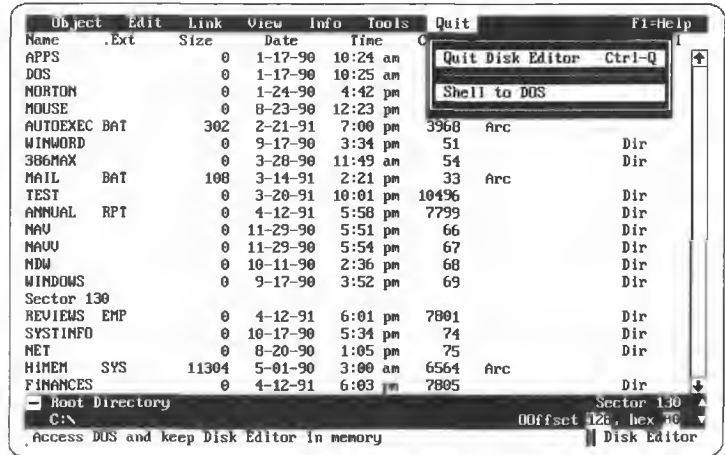
Calling DOS Temporarily

If you want to use DOS temporarily, then continue working in Disk Editor, select SHELL TO DOS. This option opens a new version of the DOS command processor in memory without terminating Disk Editor. You will see the normal DOS command prompt screen, but this reminder will also appear on the screen: Type the word `exit` at the DOS prompt to return to the Norton Disk Editor.



WARNING Do not load any TSR programs from the DOS shell (i.e., at the DOS prompt) while in Disk Editor.

Figure 2-27



Quit menu

To terminate DOS and return to Disk Editor, type `exit` at the DOS command prompt. You will return to the same setup you left: the same object, view, and cursor position, the same changes highlighted, and so on.

SHELL TO DOS is handy for checking out the changes you have made with Disk Editor. For example, if you have corrected a directory entry, you could select SHELL TO DOS, check out the directory to see if it is working all right now, then exit back to Disk Editor. You might also want to select SHELL TO DOS to locate a file through File Find or use one of the other Norton Utilities. You might need to format a diskette; you do not have to leave Disk Editor to do so if you use the SHELL TO DOS command.

Terminating Disk Editor

When you have finished examining and editing the disk and want to terminate the viewer, select QUIT DISK EDITOR or press Ctrl-Q. A dialog box asks you to confirm your decision. Then you return to whatever location you entered Disk Editor from, either the Norton Utilities main menu or the DOS command prompt.

If you have unsaved changes when you try to exit, you will see a message like “Changes have been made to sector/cluster [number].” You must either save or discard the changes before you can exit.

MENU REFERENCE

Shortcut Keys

The following provides an overview of the Disk Editor menus.

Most menu options have shortcuts—keys that you can press to trigger the function without pulling down the menu. Shortcut keys work only when no menu is showing; once the menu is pulled down, then you must use the highlighted letter, the mouse, or Enter to select the function.

The following system has been used to help you remember the keys:

Fn	Select views (View menu)
Alt	Select objects (Object menu) or menus (menu bar)
Shift-Fn	Window function (View menu)
Ctrl	Other functions (Edit, Link, Tools, Info, and Quit menus)

In most cases, the key used in conjunction with Alt and Ctrl is the first letter of the function. Thus, Ctrl-Q triggers QUIT DISK EDITOR and Alt-S triggers SECTOR on the Object menu.

Object Menu

This menu is used to select an object to edit.

Drive... Alt-D Lets you select the logical or physical drive that contains the object. The resulting dialog box lists all the available drive or disk names.

Lets you select a directory as an object. The resulting dialog box shows the directory tree for the currently selected drive.

File... Alt-F Lets you select a file as the object. The resulting dialog box lets you select the drive, directory, and filename.

Cluster... Alt-C Lets you select one or more clusters as the object. The resulting dialog box shows you the range of valid cluster numbers on the current drive and lets you enter the starting and ending cluster numbers for the desired object.

Sector... Alt-S	Lets you select one or more sectors as the object. The resulting dialog box shows you a table of sector usage for the current disk and lets you enter the starting and ending sector numbers for the desired object.
Physical sector... Alt-P	Lets you select one or more physical sectors as the object. The resulting dialog box lets you enter the coordinates of the starting sector and the number of sectors in the desired object.
Partition table Alt-A	Selects the Partition Table as the object.
Boot record Alt-B	Selects the Boot Record as the object.
1st copy of FAT Alt-F1	Selects the first FAT as the object.
2nd Copy of FAT Alt-F2	Selects the second FAT as the object.
Clipboard	Selects the clipboard as the object. Available only when the clipboard contains data.
Memory Dump Alt-M	Provides the ability to view, search, and copy data from conventional memory.
Edit Menu	This menu contains a variety of options for editing the selected object. If Read Only is on, none of the options on this menu is available except MARK.
Undo Ctrl-U	Undoes the most recent change to a byte. Up to 512 changes per sector can be remembered and undone in reverse order.
Mark Ctrl-B	Indicates that you want to mark a block for copy or fill. After selecting MARK, use the cursor keys to mark the block. Also use MARK to cancel (unmark) a block without copying or filling it.
Copy Ctrl-C	Copies the marked block to the clipboard. Up to 4096 bytes can be copied to the clipboard. The former contents of the clipboard are overwritten by COPY.

Paste Over Ctrl-V Pastes from the clipboard to the current cursor position in the object. The clipboard is not affected. The clipboard block overwrites the data in the object; changed bytes are highlighted. UNDO restores the entire block to its condition before Paste. You are warned that changes are undoable if you try to paste across a sector boundary.

Fill... Fills a block of any size with a specified character. Fill characters are written directly to disk; therefore, the block size is not limited. You are warned that changes are not undoable if you try to fill across a sector boundary.

Write changes... Ctrl-W Copies all highlighted changes from memory to the disk. After WRITE CHANGES... is selected, change highlights disappear and changes cannot be undone.

Discard changes... Undoes all highlighted changes at once.

Link Menu

This menu lets you switch to a related object quickly. The available links depend on the current object. If Quick Links is on, you can also link to a specific object by pressing Enter or by double-clicking anywhere on the current object.

File... Ctrl-F Links to the file related to the current FAT entry or directory entry.

Directory... Ctrl-D Links to the directory related to the current FAT entry or file. The appropriate directory entry is highlighted.

Cluster chain (FAT)... Ctrl-T Links to the first copy of the FAT. The FAT entry related to the current directory entry or file is highlighted.

Partition... Links to the highlighted Partition Table or the related Boot Record, whichever is more appropriate. Available only when a Partition Table is being edited.

Window... Establishes a dynamic link between two windows so that moving the cursor in one window (which must contain a directory or FAT) causes the related file to appear in the other window. Available only when two windows are in use and the active window contains a directory or FAT object.

View Menu

This menu lets you select a viewer and control window splitting.

As Hex F2	Displays current object using the hexadecimal viewer.
As Text F3	Displays current object using the text viewer.
As Directory F4	Displays current object using the directory viewer.
As FAT F5	Displays the current object using the FAT viewer.
As Partition Table F6	Displays the current object using the Partition Table viewer.
As Boot Record F7	Displays the current object using the Boot Record viewer.
(Un)Split window... Shift-F5	Creates two windows or eliminates inactive window. To accomplish the same effect with the mouse, drag the status bar up to reveal the second window.
Grow window Shift-F6	Adds one line to active window. This option is available only when windows are split. It will not function if the inactive window has reached its minimum size. To accomplish the same effect with the mouse, drag the top status bar.
Shrink window Shift-F7	Removes one line from active window. This option is available only when windows are split. It will not function if the active window has reached its minimum size. To accomplish the same effect with the mouse, drag the top status bar.
Switch windows Shift-F8	Makes the opposite window active. This option is available only when windows are split. To accomplish the same effect with the mouse, click on the opposite window.

Info Menu

This menu allows you to display detailed information about the selected object and its disk drive.

Object info...	Displays detailed information about the object. The format and contents depend on the selected object.
Drive info...	Displays detailed information about the current drive.

Map of object... Displays a map of the object's clusters in the context of all the clusters on the disk.

Tools Menu

This menu provides a variety of functions to assist in exploring and editing the disk.

Find... Ctrl-S Finds specified string in object. The resulting dialog box asks you to enter the string to search for, in hex or text mode. The search moves forward from the current cursor position and stops when a matching string is found or the end of the object is reached. The matching string is highlighted.

find Again Ctrl-G Repeats previous search using the same search string starting at the current cursor position.

Write to... Alt-W Copies the current object (as edited) to disk. The resulting dialog box asks for the target disk and a filename, starting cluster number, or starting sector number.

Print As... Ctrl-P Print a selection in either Hex, Directory, FAT, Boot Sector, or Partition Table format.

Recalculate Partition When you are editing a Partition Table, this option calculates the Relative Sector number based on the Starting Sector coordinates and the Number of Sectors based on the Starting Sector and Ending Sector coordinates. This option is available only when you are in the Partition Table view.

Compare Windows... Compares data in two windows (starting at cursor position) and positions cursor at first point of difference.

Set attributes... Changes attributes for a group of adjacent files at once. This option is available only when the directory view is active. Mark the target files as a block in the directory viewer before selecting this option. The resulting dialog box asks for the attributes to be set.

Set Date/time... Changes the date and time for a group of adjacent files at once. This option is available only when the directory view is active. Mark the target files as a block in the directory viewer before selecting this option. The resulting dialog box asks for the desired date and time.

Hex Converter...	Shows hex, decimal, and character equivalent of entered value.
ASCII Table...	Displays a table of characters and their hex and decimal equivalents.
Configuration...	Controls configuration of Disk Editor. The items that can be set are:
Read Only	Prevents changes being made to an object.
Quick Move	Speeds response time by omitting filename rewrites in the status bar.
Auto View	Selects the most appropriate viewer when objects are loaded.
Quick Links	Allows linking between related objects by pressing Enter or double-clicking.
Character filters	Controls whether or not characters are interpreted according to WordStar conventions.
	Other controls in the CONFIGURATION dialog box include:
Save	Saves configuration in NU.INI.
OK	Uses configuration settings for current session but doesn't save them on disk.
Cancel	Cancels configuration dialog box without making changes.

Quit Menu

This menu lets you exit Disk Editor, either temporarily or permanently.

Shell to DOS	Calls DOS without exiting Disk Editor. When you have finished using DOS, type <code>exit</code> to return to Disk Editor.
Quit Disk Editor Ctrl-Q	Terminates the Disk Editor and exits to the Norton Utilities main menu or DOS. If unsaved changes exist, a dialog box asks whether you want to save or discard them.

OPERATIONAL NOTES

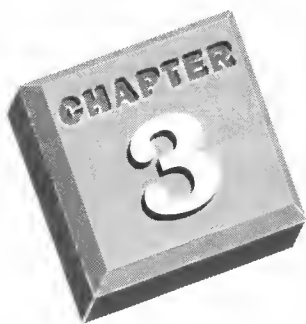
Disk Editor does work on RAM drives. Disk Editor also works on files on JOINed and SUBSTituted drives.

COMMAND LINE USE

DISKEDIT [drive:][path][filename] [/M] [/X:drives] [/W]

/M	Maintenance mode: bypasses DOS and looks directly at the disk.
/X:drives	Exclude drives from absolute sector processing (for example, /X:de f).
/W	Write-able mode (read-only off).

If /M is specified on the command line, DiskEdit will attempt to access the diagnostic cylinder of any physical hard disk. If one is present, DiskEdit will allow you to access that cylinder.



Troubleshooter

The Troubleshooter is a collection of procedures, a “recipe book” if you will, for hands-on disk and file maintenance. Over the years our excellent Technical Support team has compiled and tested these routines from the problems most commonly called in by our customers. Whereas our other product documentation features background information, extensive examples and the general philosophy behind the programs, the Troubleshooter takes you directly to the problem-solving level of disk technology.

Most (if not all) of the problems you will ever encounter on your disk can be solved automatically by the Norton Disk Doctor, which is fully described in the *User's Guide*. But if there are specialized problems for your specific system which the Norton Disk Doctor cannot solve, or if you want to take personal control of the repair process, we have supplied all the information you need in this chapter.

We have grouped the Troubleshooter procedures into four categories:

- data recovery
- directory fixes
- system area fixes
- absolute sector level fixes

Contents

Introduction	3-1
Data Recovery Procedures	
Overwritten File Fix	3-3
Recovering Cross-Linked Files	3-6
Lifting Data off a Bad Disk	3-7
Recover from DOS's RECOVER	3-14
Fixing a Trashed Root Directory	3-14
Directory Fixes	
Removing Stubborn Files	3-17
Fixing Bad Directory Entries	3-19
Directory Tree Structure Fix	3-21
Recovering Subdirectories from a Bad Directory	3-25
Removing Stubborn Directories	3-28
System Area Fixes	
Boot Record Editing	3-29
Media Descriptor Byte Editing	3-31
Fixing the DOS System Files	3-32
Absolute Sector Level Fixes	
Partition Table Editing	3-37
Fixing a Bad Track 0	3-39
Low-Level Format for an XT	3-43
Tables	
General Disk Information	3-44
File Attributes	3-45
Boot Record	3-46
Disk Types for IBMs	3-47
Disk Types for Compaqs	3-48

OVERWRITTEN FILE FIX

DOS handles overwritten files differently depending on the version of DOS. DOS 2.xx re-writes data on top of the same data clusters when you save a file over an existing file of the same name, meaning that you will lose as much data as the size of the new file. DOS 3.xx and later write the file to another part of the disk (if there is enough room on the disk) when you save a file over an existing file of the same name, with the result that the data clusters for the original file probably have not been overwritten.

There are two basic steps to recovering an overwritten file:

- finding the starting cluster of the file, and
- finding the rest of the file.

If the disk is not too fragmented, the rest of the file usually follows the starting cluster closely. If the file is small enough (for example, a 2K batch file) it may even fit in one cluster.

Step 1: Locate the starting cluster of the file.

To find the starting cluster of the file, search the disk for a unique string of text you know appears within the first 500–2000 characters of the file (depending on the disk's cluster size).

1. Select the disk (Alt-D) on which the file was overwritten as your object in DiskEdit and select all the clusters.

Pull up the Tools menu's FIND dialog box (Ctrl-S) and type into the ASCII field a string of text that is unique to the file you are searching for (a name, header or key word that would be found in that file) and press Enter. For example, if you are looking for a spreadsheet project named "Ethereal Estates", you should, if you titled the spreadsheet, search for the word "Ethereal". If you are looking for a letter to the head of General Motors, you should search for "Motors".

2. Look at the found data in the Text viewer (F3). If this data does not belong to your file, then press (Ctrl-S) to continue searching.

If the text does belong in the file, but is not the beginning of the file, write down this cluster number and then either try a new search for text that should appear earlier or look at previous clusters using PgUp.

If the cluster is the beginning of the file, check to see if it belongs to the file with which you overwrote your older file, by looking at the status line: if the cluster is allocated to another file, the filename will show on the left side at the bottom of the screen. If the cluster does not currently belong to a file, that spot will tell you that the cluster is “unallocated”. If the cluster is free, write down the cluster number and continue with Step 2. If it does belong to the older file, continue searching with (Ctrl-S).

3. If you haven't found any clusters from your file, try searching for a different text string.

Step 2: Find the rest of the file.

Unless the disk is very fragmented (meaning the files are spread out all over it, a condition you can remedy by running Speed Disk regularly, although you should not do it when attempting to recover lost data), the rest of the file should be near the starting cluster. If not, you will have to search for text from the file.

Method 1: Look near the starting cluster

1. Select the disk from the Starting Cluster you have already found to the end of the disk as your object.
2. Using the Hex viewer (F2), look at the data in each cluster. The first cluster you see will be the starting cluster in the file. From this you should be able to get some idea of what should come in the following clusters in the file.
3. PgDn (or scroll using the mouse) to the next several clusters, looking for data that you expected to come soon. When you find it, write down that cluster number and repeat the process (using the data in the newest cluster to guess at what should be in the next cluster) until you have found the proper number of clusters. Make certain that each sector you add has not already been allocated to another file, using the same method as in Step 1 (locating the starting cluster of the file).

If that failed to find the rest of the file, search the disk for text you expect to find in the next portions of the file. This will only work on text files such as word processing and database files, because others, like COM or EXE files, will be too hard to recognize.

Method 2: Search for unique text

1. Select the disk from the Starting Cluster you have already found to the end of the disk as your object.
2. Search for text (Ctrl-S) you know appears frequently in this file but not in other files. Write down each cluster that the text appears in that looks like it belongs in this file.
3. Looking at the newest cluster you found, start a search from that point for text you expect to be in the NEXT cluster, and write down the resulting cluster (if it looks proper and is not currently allocated as in Step 1).
4. Repeat (3) until you have reached the expected end of the file or until you have found the proper number of clusters. If these two things do not happen simultaneously, you have either missed some clusters or chosen too many. When you are done, proceed to Step 3.

Step 3: Write the file to disk.

Now that we have all of the clusters that formed the file before, we will write it to a file.

1. Select the first cluster of the file as your object (from your list of found clusters).
2. From the Tools menu, choose WRITE TO...
3. Give the item a name, preferably on a different drive, and hit Enter. For example, "B:MYFILE.WK1".
4. Go to the next cluster of the file (from your list), and choose WRITE TO... again. Enter the same name for the file again.
5. Choose to APPEND your cluster to that same file when the dialog box asks you.
6. Repeat (4) and (5) until you run out of clusters.
7. If you have recovered a dBASE or Lotus file, you should use File Fix to check your file's integrity at this point.

RECOVERING CROSS-LINKED FILES


When files are cross-linked, it means that two files are sharing the same cluster of data space. Think of it as two trains hooked up to the same boxcar.

The cross-linkage can take place anywhere along the chain, not just on the first or last cluster of a file's data.

Since each cluster of data can only belong to one file at a time, one of the cross-linked files will be unusable until the cross-linkage is eliminated.

Suppose you've just run NDD or DOS's CHKDSK and found a few files that are cross-linked on cluster 123. The following procedure will guide you through determining the full extent of the damage and fixing it, if NDD did not fix the cross-linked files, or if you simply want complete control over the recovery process.

It is important to follow the instructions in the order they are given.

1. Write down the names of the cross-linked files and the cluster(s) they are cross-linked on.
2. Copy all these files to another disk. If a certain file is too big, then you can copy it to another directory, though this is less preferable.
3. Load or run each file that you just copied to verify that it is either good or corrupt, and make a list of the corrupt ones. Verify the copies, not the original files.
4. Copy the good files back to your original disk. At this time your files should run correctly. 
5. If you had any corrupted files that you do NOT have on another disk, run NDD and SAVE the lost clusters it finds (it should find some or have found some before).
6. The lost clusters were saved to the root directory with the extension `._DD`. If CHKDSK was used, the files have the extensions of `.CHK`. Examine these files with DiskEdit to try to determine which belong in your corrupted files.

7. From the Directory view, write down the starting cluster of the `._DD` file that belongs in your corrupted file. Link to it in the FAT (Ctrl-T), and write down its chain length (in clusters). Return to the Directory view (Ctrl-D) and highlight the corrupted file. Link to it in the FAT (Ctrl-T).
8. The FAT chain for the corrupted file is marked in red, or with chevrons if you have a monochrome display. Move to the end of the highlighted chain and count backwards the number of clusters in the lost chain. Write over the number at this location (which points to the next cluster in the file) with the starting cluster of the lost chain. Change the clusters that HAD been in this chain but are no longer part of it to <EOF>. Write the changes, choosing to synchronize the FATs.
9. Return to the Directory view (Ctrl-D), highlight the `._DD` file, switch to Hex (F2), move to the Hex side, and enter E5. We are manually erasing this file.
10. Write the changes and return to DOS. The files should now be fully recovered. If they don't work properly, check them against another `._DD` file.

LIFTING DATA OFF A BAD DISK

When DOS will not access a disk that used to work properly, DOS considers the disk to be bad. This can be caused by a messed up boot sector, confused boot partition, or by illegal or incorrect data in the FAT or root directory. NDD will automatically fix most of these problems for you.

If NDD does not succeed in fixing your disk, then you will have to recover the necessary files through brute force. The following methods bypass DOS to find and recover your files, one by one.

There are three steps to the data recovery: Finding the starting cluster of the file, finding the rest of the clusters in the file, and writing the file out. Steps one and two consist of several methods, listed in decreasing order of ease and reliability. Each method is composed of a brief overview and then specific steps to take. We suggest you try Method 1 first, then try Method 2 only if the first fails, and so on.



NOTE: Because the disk is “bad,” you may need to use DiskEdit in the Physical Disk mode. If so, you will not be able to select as objects files or directories. Also, because we are working with a bad disk, it is very important to turn Quick Links off.

Step 1: Find the starting cluster of the file.

Method 1

Try to follow the directory structure to the directory listing so that you can look at the Cluster field for the starting cluster.

1. Try to select the disk as a Logical Disk to find the directory. If the directory structure is intact enough for you to find the file listing for your file this way, proceed to Step 2.
2. If this fails, search the disk for the file `COMMAND.COM`. Because of the way DOS stores filenames, do the search for “`COMMAND COM`”, with one space between `COMMAND` and `COM`. If you don’t have `COMMAND.COM` in your root directory, search for “`CONFIG SYS`”, with two spaces this time.
3. Switch to the Directory view (F4) .
4. If the file is in this directory, write down the number that is in the Cluster field and the file size and go to Step 2.

Otherwise, go to the cluster number corresponding to the directory that holds the file and repeat the process until you find the directory listing.

Method 2

If the directory structure failed, search all of the disk for the filename in an attempt to find the file’s directory.

1. Select the entire disk as your object.
2. Search (Ctrl-S) for the filename. Note: the filename is not entered in the format you usually see it in. Instead, of `NAME.EXT`, enter it in the search field as “`NAME EXT`”, with no period between the name and extension, and with spaces padding the `NAME` section so that it totals eight characters.

3. When the text is found, hit F4 to view it in the Directory viewer. If the screen is gibberish, search for another occurrence of the filename (Ctrl-G).
4. If the Directory view shows a directory, write down the starting cluster number and the file size of the file from those fields.

Method 3

Failing the above, we will try to find the starting cluster by searching the disk for “.. ”, under the assumption that perhaps the filename is slightly garbled or misspelled. This will show you all the files on the disk.

1. Search for “.. ” (2 dots, 6 spaces). ((Ctrl-S) will get you to the FIND dialog box.) This will locate all the subdirectories on your disk.
2. When a selection appears, press F4 to view it as a directory. If it appears to be a directory, write down the starting cluster number and size of the file you are after (or print the screen using Shift-Print Scrn). If the file you are after does not appear in the listing, continue searching for more subdirectories (Ctrl-G).

Method 4

The last resort for finding the starting cluster of the file is to search the disk for a unique string of text you know appears within the first 500–2000 characters of the file. We do this because the directory entry is probably scrambled beyond recognition, but the file may still be intact.

1. Pull up the FIND dialog box (Ctrl-S) and type into the ASCII field a string of text that is unique to the file you are searching for (a name, header or key word that would be found in that file) and press Enter. For example, if you are looking for a spreadsheet project named “Ethereal Estates”, you should, if you titled the spreadsheet, search for the word “Ethereal”. If you are looking for a letter to the head of General Motors, you should search for “Motors”.

2. Look at the found data in the Text viewer. If this data does not belong to your file, then press Ctrl-G to continue searching.

If the text does belong in the file, but is not the beginning of the file, write down this cluster number and then either try a new search for text that should appear earlier or look at previous clusters using PgUp.

If the cluster is the beginning of the file, write down the cluster number and continue with Step 2.

3. If you haven't found any clusters from your file, try using a different text string to search for.

Step 2: Find the rest of the clusters in the file.

Method 1

Check the file size you wrote down from the directory information (unless you used Method 4 of Step 1). Compare this to the cluster size of the disk. This can be found by selecting the DRIVE INFO option from the Info menu when in Logical mode. (Most hard disks have 2048-byte clusters, while most floppies have 512-byte clusters. Hard disk partitions larger than 32MB usually have 4096 bytes per cluster.) If the cluster size is larger, you have found the entire file and should proceed to Step 3. Otherwise, determine how many clusters you need by dividing the file size by the cluster size and rounding up. Write this down and continue with Method 2 below.

Method 2

Since we need more clusters, we will look in the FAT to see where it thinks they are, and then we will check the number of clusters the FAT claimed were there for validity.

1. Try to select the FAT as your item. You will not be able select it if you had to access the disk in Physical mode, in which case you will have to search for the FAT. It usually begins, on hard drives, with F8 FF FF (hex), and is at the beginning of the disk. Remember, if you are trying to recover data from a second partition, search the beginning of that partition. When the program finds the text, switch to the FAT view and see if the numbers have a reasonably ordered progression. If not, repeat the search again (Ctrl-G).



NOTE: If you cannot find or use the FAT, or if it is too corrupted, and if you had IMAGE in your AUTOEXEC.BAT, try using the IMAGE.DAT file on the disk. IMAGE.DAT contains a copy of the FAT, which is current as of the last time you ran IMAGE. This file begins with the characters "PNCIHIBK", so performing a search for this text may turn it up. The FAT portion of this file still begins with F8 FF FF, and is probably at about the fourth sector. Turn on the FAT viewer and PgDn until it shows legal values progressing from low numbers.

2. When you have found the FAT, PgDn or scroll to the starting cluster number you found before. Note: you are not trying to find the number on the top of the screen, although those numbers are usually near the cluster numbers they represent. Instead, you are looking for the cluster number as displayed on the status line.
3. Follow the FAT chain, writing down cluster numbers as you go. The starting cluster should either show <EOF> or have the number for another cluster. If it says <EOF>, you are done. If it has another cluster number, go to that cluster (in the FAT) and check it, writing it down in the process. Repeat this until you reach an <EOF>.
4. Compare the number of clusters the FAT found with the number you need (from your calculations in Method 1 of Step 2, above). If they are not the same, you will have to try Method 3 below. If they are the same, go to Step 3, writing the data.

Method 3

If more clusters are needed and the FAT cannot help, we will check the next several clusters for likely data. This assumes that the disk has little fragmentation.

1. Select the disk from the Starting Cluster you have already found to the end of the disk as your object.
2. Using the Hex viewer (F2), look at the data in each cluster. The first cluster you see will be the starting cluster in the file. From this you should be able to get some idea of what should come in the following clusters in the file.
3. PgDn (or scroll using the mouse) to the next several clusters, looking for data that you expected to come soon. When you find it, write down that cluster number and repeat the process (using the data in the newest cluster to guess at what should be in the next cluster) until you have found the proper number of clusters.

Method 4

Our last resort for finding the rest of the file is to search the disk for text you expect to find in the next portions of the file. This will only work on text files such as word processing and database files, because others, like COM and EXE files, will be too hard to recognize.

1. Select the disk from the Starting Cluster you have already found to the end of the disk as your object.
2. Search for text (Ctrl-S) you know appears frequently in this file but not in other files. Write down each cluster that the text appears in that looks like it belongs in this file.
3. Looking at the newest cluster you found, start a search from that point for text you expect to be in the NEXT cluster, and write down the resulting cluster (if it looks proper).
4. Repeat (3) until you have reached the expected end of the file or until you have found the proper number of clusters. If these two things do not happen simultaneously, you have either missed some clusters or chosen too many. When you are done, proceed to Step 3.



NOTE: These last two methods (3 and 4) are last not only because they are the slowest and hardest, but also because these are the least certain methods for deciding which data to put in the file. If you have several spreadsheet files, all of the data may look alike, but the FAT chain will know which data belongs where.

Step 3: Write the file to disk.

Now that we have all of the clusters that formed the file before, we will write it to a file. Because the drive you are working on is messed up (and because writing to it may overwrite other files you may wish to recover), you should write this new file onto a different disk (perhaps a floppy).

1. Select the first cluster of the file as your object (from your list of found clusters).
2. From the Tools menu (Alt-T), choose WRITE TO...
3. Give the item a name, preferably on a different drive, and hit Enter. For example, B:MYFILE.WK1.
4. Go to the next cluster of the file (from your list), and choose WRITE TO... again. Enter the same name for the file again.
5. Choose to APPEND your cluster to that same file when the dialog box asks you.
6. Repeat (4) and (5) until you run out of clusters.
7. If you have recovered a dBASE or Lotus file, you should use File Fix to check your file's integrity at this point.

You are done recovering this file. Repeat these steps for all the files you wish to recover.

RECOVER FROM DOS'S RECOVER

DOS's RECOVER program is designed to help the user get files back after a partial format or after the root directory has been destroyed. RECOVER simply creates an entry in the root directory for every chain in the FAT (every file), obliterating the previous root directory and destroying the filenames. The idea behind the program was that if you can't access any of your files, this would at least give you all of your files, and let you decide which was which.

RECOVER is a dangerous program, and should be removed from your hard disk to prevent it from being run accidentally. The Norton Disk Doctor and Norton UnFormat will properly fix the problems that RECOVER is designed to address, and should be used instead of it.

If you have already run RECOVER, your files are still intact. The FAT has not been changed. The only change is that the root directory has been replaced, or, from our point of view, trashed. The procedure to recover from this is the same as that to recover from a trashed root directory, so use that procedure (Fixing a Trashed Root Directory) to fix this problem.

FIXING A TRASHED ROOT DIRECTORY

When your root directory appears to be trashed (when DIR returns gibberish or maybe "File not found"), the problem may be due to bad directory entries. Try the procedure for Fixing Bad Directory Entries to fix those first.

If the root directory is truly bad, most of the files should be fully recoverable. The subdirectories can be recovered intact as long as the damage was confined to the root directory, simply by creating new entries for them. After that, any files that had been in the root directory will show up as lost chains on the NDD or CHKDSK reports.

STEP 1: Locate root level directories.

1. Run DiskEdit and select the entire disk in Cluster mode as your object.
2. Search for ".. " (2 dots, 6 spaces). ((Ctrl-S) will get you to the Find dialog box.) This will locate all the subdirectories on your disk.

3. When the text is found, hit F4 to view it in the Directory viewer. If the screen is gibberish, search for another occurrence of the text (Ctrl-G).
4. If the found text displays as a proper directory, check for root level directories (check the “..” entry for a Starting Cluster Number of 0). Write down current cluster number (which should be the same as in the “.” entry) and the original name of the subdirectory (or at least a temporary name). Do not worry about subdirectories that are not off of the root directory (as indicated by the “..” entry Starting Cluster Numbers) because they will be reactivated as children of the root level directories.

STEP 2: Clear and set up the root directory again.

1. Clear out the root directory except for the first two directory entries, which should be for the two system files. If you are working on a non-bootable partition, clear out the first two entries also. Use the Hex view (F2) and the FILL command (from the Edit menu) to fill the directory (after the first two entries) with 00 hex.
2. If the disk you are working on is your boot disk, select the Boot Record as your object, switch to the Hex view F2, and look for two filenames (in all upper case, without periods) something like “IBMBIO COMIBMSYS COM”. Note that there is no space between the extension of the first file and the beginning of the second. You may have to hit PgDn to get to them. Other common names will use SYS as the extension. Write down the filenames and (back in the root directory) rename FILE0000.REC and FILE0001.REC to these two filenames (in order).
3. Use the Directory view screen to create a new root directory entry for each of the root level directories (simply type over the empty fields or the “Unused directory entry” banner). Since the FAT is fine, this will immediately activate most of the disk.
4. Use NDD or CHKDSK to convert the remaining lost clusters to files: these will be the original root directory files.

5. Search the ._DD or .CHK files for any directories. Use File Find and search for the text of “.. ”. Hopefully, no additional directories should be found. If an additional directory is found, view it from DiskEdit in Directory mode, and, if necessary, create a root directory entry for it on top of the file entry that NDD or CHKDSK created for it.



NOTE: This section assumes you have run Image regularly, for instance in your AUTOEXEC.BAT. If not, you will have to look at the files one-by-one to determine their names.

STEP 3: Rename root directory files.

1. Use DiskEdit to locate the most recent file (search the disk for “PNCIHIBK”, which is the header for that file). Switch to Directory View (F4), PgDn until you see a copy of your Root Directory, and use the time and date of the IMAGE.DAT directory entry to choose and verify the most recent copy if DiskEdit found the header several times.
2. Split the DiskEdit screen into two windows with IMAGE.DAT on top and the newly set up root directory in the bottom window. The root directory should contain the two system files properly renamed, a number of directories, and several ._DD or .CHK files depending on whether NDD or CHKDSK was used.
3. Move the highlight to the Starting Cluster Number of the first file to rename in the bottom window—the root directory.
4. Use the Hex Converter to convert the number into hex and rotate the two bytes to achieve backward order.

Example:	7,146	1BEA	EA1B
	Decimal	Hex	Rotated

5. Change the active window to the top window (IMAGE.DAT file), activate the FIND option (Ctrl-S) and search for the rotated hex number (stay in the Directory view).

6. The highlight bar will move to the first match. Double check the Starting Cluster Number and size of both files to make sure it is actually the same file.
7. Change the active window to the bottom window (root directory) and type over the FILE???._DD or FILE???.CHK filename with the proper one.
8. Mark the filename in the top window with a first character "." (2E hex) to identify it as found.
9. Some files may not exactly match because they might have been edited or moved since the IMAGE.DAT "snapshot" was taken. By using the approximate file size and by viewing the contents of the few remaining files you can easily match them up.

REMOVING STUBBORN FILES

Sometimes you can't remove a certain file by simply using the DOS DELETE command. It may be that the file in question does not use the standard ASCII character set (or it may have a lower-case character in the filename, which DOS filters out and cannot process). Alternatively, the read-only attribute may be turned on, preventing you from erasing the file.

Method 1

Try clearing the file's attributes, using the File Find program, and then deleting the file.

Example:

```
a:\> File Find c:\progs\workin.doc /CLEAR  
a:\> DEL c:\progs\workin.doc
```

The above example assumes that you wish to remove the file c:\progs\workin.doc and that the Norton Utilities are in your path.

Method 2

Rename all files which have names that are similar to your file. For instance, if the file is called WORKIN.DOC (there may or may not be any hidden characters at the end), rename the other files so that you can safely run "ERASE wo*.d*". Using wildcards you can delete this file without having to re-create those strange extra characters.

Method 3

If the above method failed, it may be that your directory is corrupt and DOS simply cannot recognize the file. We can still delete the file from DOS by manually renaming it to a normal name.

1. Start DiskEdit and choose the directory where the problem file is found as your object.
2. If you are not already in Directory view, switch to it (F4).
3. Place the highlight bar over the first character of the filename you wish to delete and rename the file to something else. Write the changes (Ctrl-W).
4. Repeat the above for all the problem files, and then exit to DOS.
5. Move to the directory of the problem file and delete the file.

Method 4

If the above methods didn't work, we will delete the file manually by putting the "erased file" character at the beginning of the filename, and then cleaning up the lost clusters that had been associated with that file.

1. Start DiskEdit and choose as your object the directory where the problem file is.
2. Place your highlight bar over the first character of the filename.
3. Press Alt-F5. You will see that the file now begins with the sigma character. This is how DOS designates a file as deleted.
4. Repeat this procedure for each problem file, and then write the change out (Ctrl-W) and exit to DOS.
5. Next run NDD and diagnose the disk that had the stubborn files. NDD will probably find lost chains. This is normal at this point. You should have NDD fix the lost chains and then delete them.

FIXING BAD DIRECTORY ENTRIES

Use this procedure if you notice that:

- Some files have disappeared from an otherwise normal directory, but were not actually erased.
- CHKDSK reports "Tree not processed past this point".
- There are corrupt-looking filenames in your directory.

When DOS scans a directory, it reads all entries until it encounters the first blank entry, which is one that has a zero for its first character. Sometimes, however, valid directory entries will be placed after a blank directory entry, which makes them unreachable by DOS. Another problem that can happen with directories is that DOS will sometimes put an illegal character in a directory entry, or turn a directory entry into garbage.

The solution to this problem is to put a valid directory entry in every blank directory entry preceding the list of valid entries. DiskEdit makes this simple, because it reports a blank directory entry as an unused directory entry, and a corrupted directory entry is shown in red, or with chevrons if you have a monochrome display.

Many of the problems will be fixed by DiskEdit when it re-writes the directory entry if you force it to. Failing that, you will have to replace the file entry with a "dummy" entry, which is equivalent to an erased file.

Step 1: Rewrite the entry.

1. Run DiskEdit and select as your object the directory with the bad directory entries. Switch to Directory view (F4) if you are not already in it.
2. Find the last valid filename in the directory by using the Down Arrow and/or the PgDn key. As you page through the directory, look for any entries displayed in red, or in chevrons if you have a monochrome display (which signifies that it is not a proper directory entry).

3. Tab over to the time and date fields for the invalid record. If you see any values that are clearly illegal (such as a strange date or a bizarre filename), fix them. Otherwise, simply change the date and/or time entries.
4. Write the changes (Ctrl-W), exit to DOS, and look for the problem. If the problem is still there, go to Step 2. Otherwise, you are done.

Step 2: Erase the entry.

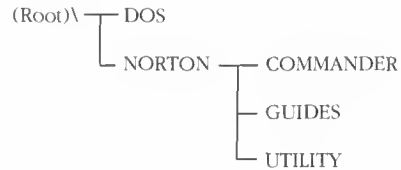
1. Run DiskEdit and select as your object the directory with the bad directory entries. Switch to Directory view (F4) if you are not already in it.
2. Find the last valid filename in the directory by using the Down Arrow and/or the PgDn key. As you page through the directory, look for any entries marked "Unused directory entry" or displayed in red, or with chevrons if you have a monochrome display (which signifies that it is not a proper directory entry).
3. If you found an unused directory entry or an invalid directory entry in any position above the last valid filename, you will need to remove them. Use the up and down arrow keys to put the cursor on the line containing the invalid entry that you want to remove.
4. Press Alt-F5. If there are additional invalid entries, repeat the process until the entries are erased.
5. Write the changes (Ctrl-W) and exit to DOS.
6. Run NDD and diagnose the disk that you just worked on. NDD will probably find lost chains. Have it fix and delete them.

DIRECTORY TREE STRUCTURE FIX

Subdirectories in DOS are actually files that have directory entries within them. Each entry contains information about a file's name, size, creation date, creation time, attributes, and starting location.

The examples below show a root directory and a few of its subdirectories. We will navigate from the root directory to the \NORTON directory to the \NORTON\UTILITY directory.

DIRECTORY TREE STRUCTURE



ROOT DIRECTORY

						Attributes					
Filename	Ext	Size	Date	Time	Cluster	Arc	R/O	Sys	Hid	Dir	Vol
IBMBIO	COM	7196	5/23/86	4:48 pm	2	Arc	R/O	Sys	Hid		
IBMDOS	COM	27760	5/14/85	12:02 am	6	Arc	R/O	Sys	Hid		
DOS			10/16/87	3:11 pm	21						Dir
NORTON			10/16/87	3:15 pm	24						Dir
AUTOEXEC	BAT	204	11/24/87	3:30 pm	50						
COMMAND	COM	23098	11/27/85	6:22 pm	51						
CONFIG	SYS	73	11/24/87	3:30 pm	97	Arc					

This is the root directory. Notice that the \DOS and \NORTON directories are listed like any other file. What separates a directory from a file is that the "Dir" attribute (on the far right) is set on, and there is no size. Everything else is the same.

The number in the Cluster column points to the location on the disk where the file or directory lies, so let's move on to cluster 24, where we will find the \NORTON directory.

Cluster 24 \NORTON DIRECTORY

Attributes											
Filename	Ext	Size	Date	Time	Cluster	Arc	R/O	Sys	Hid	Dir	Vol
.			10/16/87	3:15 pm	24					Dir	
..			10/16/87	3:15 pm						Dir	
COMANDER		10/16/87		3:17 pm	36					Dir	
GUIDES			10/16/87	3:17 pm	37					Dir	
UTILITY			10/16/87	3:15 pm	38					Dir	

This is the \NORTON subdirectory. Notice that the first two entries are called "." (one period) and ".." (two periods). These two entries are found in all subdirectories, always as the first two entries. DOS uses these entries as signposts to tell where a subdirectory is currently located, and where it came from.

Notice that the "." entry points to itself—its starting cluster is the same as the actual cluster location of the subdirectory on the disk. In this case, we are in cluster 24 and the "." entry points to cluster 24. DOS uses this file for diagnostic purposes and so will we.

The ".." entry points to the parent directory, which is where it came from. Notice that the ".." entry has a blank for a starting cluster number. A blank means that it came from the root directory.

Let's move on to cluster 38, the \NORTON\UTILITY directory:

Cluster 38 \NORTON\UTILITY DIRECTORY

Attributes										
Filename	Ext	Size	Date	Time	Cluster	Arc	R/O	Sys	Hid	Dir
			0/16/87	3:16 pm	38					Dir
			10/16/87	3:16 pm	24					Dir
MARY		928	11/09/87	4:09 pm	6643	Arc				
ASK	EXE	1184	5/15/87	4:00 pm	1881	Arc	R/O			
BEEP	EXE	5838	9/25/87	7:32 pm	1882	Arc	R/O			
DS	EXE	26674	5/15/87	4:00 pm	1888	Arc	R/O			
DT	EXE	17784	5/15/87	4:00 pm	1902	Arc	R/O			
FA	EXE	7348	5/15/87	4:00 pm	1911	Arc	R/O			
FF	EXE	7942	5/15/87	4:00 pm	1915	Arc	R/O			
FI	EXE	15180	5/15/87	4:00 pm	1919	Arc	R/O			
FR	EXE	12500	5/15/87	4:00 pm	6713	Arc	R/O			

This is the \NORTON\UTILITY subdirectory. This directory has the same format as any other directory, but notice that the “..” entry points to cluster 24, which is the \NORTON directory—the parent directory.

Now that you understand how directories work, we can use the following procedure to trace through your directory tree and verify that it is properly constructed.

Step 1: Locate subdirectories.

1. Start DiskEdit and choose Clusters on the problem disk as your object. This will prompt you for a range of clusters.
2. For the starting cluster, enter the lower number listed on the “Valid cluster numbers are...” line. Press Tab, then enter the higher number for the ending cluster. Press Enter.
3. Using FIND on the Tools menu, search for “.. ” (2 periods, 6 spaces). This appears in every subdirectory and is unlikely to appear anywhere else. When the program finds a subdirectory, it will automatically display it, but you may have to press the F4 key to view it in the proper format.

If the cluster found is a directory, there will be a single dot (.) for the first entry and a double dot (..) as the second entry. If this is the case, print the screen on your printer using the Shift-Print Scrn or Print Scrn keys on your keyboard. If it doesn't have these two entries, ignore it and move to the next step.

4. Continue the search (to find other directories); Ctrl-G will look for the next occurrence. If it finds other directories, print these also. When you have reached the end of the disk, proceed to Step 2.

Step 2: Recreate the root directory's subdirectory entries.

1. Look at your subdirectory printouts. The first entry of a subdirectory is always ".". Circle the number that appears under Cluster for this entry.
2. The second entry of a subdirectory is always "..". This refers to the parent directory. If the directory is a subdirectory off of the root directory, there should be a blank space under Cluster. If there is a number under Cluster it means that the parent to this directory is also a subdirectory. Determine the name of the parent directory by looking through your other printouts for the directory name listed in the Filename column. Write the name of the parent directory on the printout.
3. Repeat the above for all the printouts.
4. Now, from the pile of subdirectory printouts, select the printout of a directory that you want to check.
5. In DiskEdit, select the parent directory of the lost subdirectory as your object. Switch to the Directory view (F4) if necessary.
6. Under the Filename column, look for the name of the subdirectory that you want to fix. You may need to press PgDn to see more of the directory names.

Under the Cluster column, you should see the same number that is circled on the printout (the cluster number for the single-dot entry).

7. Make certain that all of the attributes other than Dir are off for the subdirectory. You can toggle Attributes with the Spacebar. The Dir attribute should be on. Turn it on if it is off.
8. Write the changes (Ctrl-W).
9. If you have any other directories to fix, select another printout and repeat the steps above until you are done.

Things to Remember:

Make sure that each subdirectory entry has a cluster number.

Make sure that the cluster number listed is the cluster location of the directory. Verify this by actually going to the directory (or to the cluster).

Make sure that all directories (except for the root) have valid "." and ".." entries as the first two entries.

Check to make sure that the cluster number for "." is the same as the cluster that the directory is in.

**RECOVERING
SUBDIRECTORIES
FROM A BAD
DIRECTORY**

This procedure is to be used for recovering subdirectories fully intact from their damaged parent directory.

Directories are simply a special type of file, but unlike files, directories contain information that refers to other directories. Except for the root directory, all directories contain information that tells what the parent directory is, and if they have subdirectories, they will have additional information for each one.

This directory information is called a pointer, because it points to another location. If a directory goes bad, we can adjust the pointers of any subdirectories to point back to a safe place (we will use the root directory). By doing this, you can recover all the files in your subdirectories quickly and easily.

Step 1: Locate the bad directory.

1. Run DiskEdit and select as your object the parent directory to the subdirectory you are trying to recover. This could be the root directory.
2. The directory listing will appear on the screen. Switch to the Directory view if necessary with F4. Locate the name of the bad directory and look under the Cluster column. Write this number down. You will need this number later.

Step 2: Find the subdirectories

1. Select as your object the Clusters on the disk with the bad directory. A message stating "Valid Clusters are..." will list a range of clusters on the disk. Type the lower number for the Starting Cluster and press Tab. Type the higher number for Ending Cluster and press Enter.
2. Use FIND on the Tools menu to search for ".. " (2 periods, 6 spaces). This appears in every subdirectory and is unlikely to appear anywhere else. When the program finds a subdirectory, it will automatically display it, but you may have to press the F4 key to view it in the proper format.
3. Look at the second entry in the directory listing, the double-dot (..) entry. If the number under the Cluster column for this entry does not match the number that you wrote before, continue the search (Ctrl-G) until you find a matching directory or reach the end of the disk.
4. Print the screen on your printer by using the Print Scrn key (or Shift-Print Scrn), and then continue the search (Ctrl-G); repeat this process until you reach the end of the disk.

Step 3: Adjust the directory pointers.

1. Select one of the printouts from the last step.
2. The first entry in any subdirectory will be a single dot (.). Circle the number that appears under the Cluster column for this entry.
3. Select that cluster as your new object, and type the number you just circled as your starting cluster.
4. The directory that appears on the printout should be the same one as the one that appears on the screen. Switch to Directory view with F4 if necessary.

The second entry in any subdirectory will be a double dot (..). Move the cursor to the number that appears under the Cluster column for this entry, and delete the number. This changes the directory pointer from pointing to the bad directory to pointing to the root directory.

5. Save the changes (Ctrl-W), and repeat Step 3 until you have recovered all of your directories.

Step 4: Create directory entries in the root directory.

1. Select Sector for your object. A dialog box for your choice will appear, with a chart outlining Sector Usage near the bottom of it. Locate the sector range for "Root dir", entering the lower number as your Starting Sector and the higher number as your Ending Sector. Remember to Tab between the fields.
2. The root directory will appear on the screen. If necessary, press the F4 key to view it in its proper format, and use the Down Arrow key to move through the directory. Place the cursor on the first "Unused directory entry" or the first entry displayed in red, or with chevrons if you have a monochrome display (which denotes an invalid entry).
3. Select one of the printouts and enter the name that belongs to that directory in the filename field. If you don't know the name of the directory, make one up. You can use the Norton NCD program later to rename any directories.
4. Tab over to the "size" column and make the entry a zero.
5. Tab over to the "Ext" column and clear it out by pressing the Spacebar three times.
6. Enter the cluster number you circled on the printout (the entry for the single-dot entry) as the Cluster entry in that field for the file you just named.
7. Turn all of the attributes for this entry off, except for the Directory entry, which you should turn on. Toggle the attributes with the Spacebar.
8. Repeat the above steps for each of your printouts, and then write the changes to disk (Ctrl-W).

Step 5: Finishing touch

If you plan to recover any files from the original directory, you should do that before continuing here. Since we have moved the directories, we may have lost chains and invalid entries, which we will now clean up.

Run NDD, and diagnose the disk you just fixed. Allow NDD to fix the invalid entries and the lost chains. Choose to delete the lost chains.

REMOVING STUBBORN DIRECTORIES

A directory cannot be removed unless it is completely empty of all files. There are a few obstacles that can prevent us from clearing out the files in a directory:

Files cannot be erased if the read-only, hidden, or system attribute is turned on. Hidden files cannot be seen from DOS, so the directory may appear to be empty. You can use the Norton Utilities File Find program to turn these attributes off.

Sometimes a directory name can be corrupted, usually with characters that cannot be entered from the keyboard, or with invalid characters such as spaces or lower case characters.

Use this procedure if you have a directory that you cannot remove from DOS.

Step 1: Clear out the directory.

The first thing to do is make certain all the files have been erased.

1. Run File Find on the directory with *.* /CLEAR as the command-line parameters. For example, for the DOS directory of your C: drive, you would type:

```
File Find C:\DOS\*.* /CLEAR.
```

This will clear the file attributes, making the files visible and erasable.

2. Erase all of the files that appeared at this point (DEL *.* from the proper directory).

3. Move to the root directory, or any directory above the current directory to remove the directory. Type RD, then the full pathname of the directory to be removed. For example, to remove the C:\DOS directory, you would type:

```
RD C:\DOS
```

If this did not successfully remove the directory, we will have to do it manually.

Step 2: Remove the directory entry.

1. Run DiskEdit and select as your object the drive and parent directory of the directory you wish to remove. For example, if you wished to remove C:\BUBBLE, select the root directory. If you wished to delete C:\BUBBLE\SOAP, select C:\BUBBLE as your object.
2. Switch to the directory view if you are not already in it (F4). Under the Name column, find the directory to be removed. You may need to use the PgDn key. Highlight the directory name.
3. Press Alt-F5.
4. Write the changes (Ctrl-W) and return to DOS.

Check to see that the subdirectory is gone. If not, repeat the process carefully.

Now we have to clean up the space allocated to the directory before, which is now in the form of lost chains.

Start NDD and use it to diagnose the disk that had the stubborn directory. NDD will probably find some lost chains. Have it fix and delete them.

BOOT RECORD EDITING

The Boot Record (not to be confused with the Master Boot Program) is the beginning sector (Sector 0) of a logical disk.

The Boot Record stores important information about the drive such as the type and version of DOS used for partitioning and formatting, and the dimensions of the logical drive.

On a floppy disk, the Boot Record is the physical and logical beginning of the disk. On a hard disk, the Boot Record is the beginning of a specific partition (logical disk drive), but it is not the beginning of the physical disk. Instead, the beginning of the physical disk holds the Master Boot Program and the Partition Table. The Master Boot Program is the first program the computer runs when you boot-up the computer. This and the Partition Table let the computer know that one or more logical drives are installed and may be bootable.

The Partition Table points to the Boot Records of all logical drives located on that physical disk. For example, if you have a 60MB disk partitioned in a 30MB drive C: and a 30MB drive D:, you would have two Boot Records located on the physical disk, and the partition table would store the absolute sector location of both these Boot Records as well as other physical dimensions about these logical drives.

If you suspect that some information may be corrupt or inaccurate in a Boot Record, use the following procedure.

1. Start DiskEdit and select as your object the disk to be edited. Select your object (again), this time the Boot Record.
2. Edit the Boot Record information as needed using the "General Disk Information" tables at the end of this section. Some of the current interpreted values can be seen using the Info menu, with the Boot Record as your object.
3. Write the changes with Ctrl-W.

The computer will have to be rebooted before the changes will take effect (they are only read at boottime).



NOTE: There are some versions of DOS and third party partitioning software that allow you to use a partition greater than 32MB. For example:

- DOS 4.xx and higher
- Compaq DOS 3.31 and higher
- Wyse DOS 3.21 and higher
- Zenith DOS 3.3 and higher
- Disk Manager partitioning software
- SpeedStor partitioning software
- VFeature Deluxe partitioning software
- DR DOS 3.4 and higher

MEDIA DESCRIPTOR BYTE EDITING

The Media Descriptor byte is located in three different places on your disk (the Norton Disk Doctor will automatically find and correct problems with the media descriptor byte):

- Sector 0, Offset 21 of the Boot Record
- Sector 1, Offset 0 of FAT 1
- Sector 1, Offset 0 of FAT 2.

Different versions of DOS use either of the first two locations for disk information. The second copy of the FAT is never used by DOS (except for CHKDSK).

Use the following procedure to view or edit this byte in the Boot and FAT areas:

1. Start DiskEdit and choose the drive you wish to edit as your object. Select Boot Record from the Object menu (again).
2. Highlight the Media Descriptor Byte field at offset 21, hex 15. This is shown at the bottom right of the screen.
3. Compare the byte to the values in the chart below.
4. If the value is incorrect, change it and write the changes (Ctrl-W).

5. Choose 1ST COPY OF FAT from the Object menu. Change from the FAT view to the Hex view (F2).
6. Select the byte at offset 0, hex 0 (the offset is shown on the bottom left on the status bar).
7. Check the chart below to see if the byte is correct. If not, edit it accordingly and write the changes (Ctrl-W).
8. Choose 2ND COPY OF FAT from the Object menu and repeat steps 6 and 7.

The changes will not take effect until you reboot.

Disk Type	Media Descriptor Byte
5 ¹ / ₄ " Diskette, 160K, Single-Sided, 8 Sectors	FE
5 ¹ / ₄ " Diskette, 180K, Single-Sided, 9 Sectors	FC
5 ¹ / ₄ " Diskette, 320K, Double-Sided, 8 Sectors	FF
5 ¹ / ₄ " Diskette, 360K, Double-Sided, 9 Sectors	FD
5 ¹ / ₄ " Diskette, 1.2M, High-Density, 15 Sectors	F9
3 ¹ / ₂ " Diskette, 720K, 9 Sectors	F9
3 ¹ / ₂ " Diskette, 1.44M, 18 Sectors	F0
Hard Disk	F8

FIXING THE DOS SYSTEM FILES

Use this procedure if your disk works properly, but it is no longer bootable. Do not use this procedure if you are trying to upgrade to an incompatible version of DOS. For example, if you want to go from a 2.x to a 3.x version of DOS, you will need to reformat the disk with DOS' FORMAT.COM program. (You do not have to reformat the disk if you upgrade to a DOS version 4.x or higher.)

In order to make your disk bootable, first try to use the Norton Disk Tools program called Make a Disk Bootable or the DOS SYS command to transfer the system files to the disk. Everything that we are doing manually here is done automatically by the Make a Disk Bootable program.

A hard disk may become unbootable if the partition table has been modified to indicate "NO" under the section called Boot (for the bootable drive only). Use the procedure for Partition Table Editing if you suspect that this has been changed.

If these procedures are ineffective or if you get the message “No room for system files” when you run SYS C:, continue with the following procedure.



NOTE: Only A: and C: can be bootable. Because A: refers to floppy drives, this procedure is for fixing drive C:.

Step 1: Free up any conflicting clusters.

The DOS System files must physically be the first two files on the disk. This procedure moves any other files that are currently there to make room for the system files.

1. Boot from a floppy with the same version of DOS as on the disk you are fixing, and start DiskEdit.
2. Select as your object the FAT on drive C:.
3. If you have Auto Views turned off (if you are not currently in the FAT viewer), switch to the FAT viewer (F5).
4. Look at clusters 2 through 29.

The highlight should be on cluster number 2, although the number highlighted should not be 2. The cluster number is shown on the status bar at the bottom of the screen, to the right of the filename to which that cluster belongs. Usually, Cluster 2 will point to Cluster 3.

If a cluster points to 0, it is currently marked as unused. Once we have moved all the necessary files, clusters 2 through 29 will all say 0.

5. On a piece of paper, write down the filenames (including full path) that appear when you highlight each of the clusters between 2 and 29. If just a directory name shows, write down that directory name. Most of these names will be repetitions, so only write those down once.
6. Exit to DOS.
7. Copy each of these files to another disk manually.
8. Erase each of these files from the hard disk.

9. If any directory names were on your list, copy all of the files from that directory to another disk, delete those, and remove the directory (with `RD directory_name`).

At this point, SYS C: should work. If it does, recreate your directories and copy all of the files back to the hard drive.

The following steps are primarily for general interest. SYS C: will automatically take care of all of the following, once space has been made for the files as in Step 1, above.

Step 2: Determine the names of the two system files.

In PC-DOS and many types of MS-DOS, the two system files are called `IBMBIO.COM` and `IBMDOS.COM`. But in some types of MS-DOS, `IBMBIO.COM` is called `IO.SYS` and `IBMDOS.COM` is called `MSDOS.SYS`.

Name your system files according to the type of DOS you are going to reinstall on to your disk. To verify what your system files are named, view the directory of the bootable disk using the following procedure.

1. Start DiskEdit and choose a drive with the system files as your object.
2. Switch to the Directory view of the root directory and write down the names and sizes of the first two files exactly as you see them. Be sure to include their extensions.

Step 3: Adjust the root directory.

In addition to having to be physically the first files on the disk, the DOS system files must often also be the first files listed in the root directory.

1. Select as your object the root directory of the drive you wish to make bootable.
2. Make sure that the first two entries are free, in other words, that they are either (1) old system files, (2) erased files, or (3) unused directory entries. Otherwise, you may be looking at the root directory of the wrong disk drive. If this is not the case, copy the two files that are there to another disk, delete them from this disk, and continue.

3. To find the Cluster size on the drive, choose the Info menu (Alt-I), and select the DRIVE INFO option. Multiply the Bytes per Sector by the Sectors per Cluster to get the Bytes per Cluster.
4. Create the directory entries (we are doing this manually, where SYS and DOS would do it automatically):
 - a) Return to the Root Directory screen.
 - b) Type over the first entry's name with the first system filename you wrote down from your bootable disk.
 - c) Tab to the extension field and type in the extension.
 - d) Tab to the file size field and type in the file size from the system file.
 - e) Under the Starting Cluster field, input 2. This is the first filespace spot on the disk.
 - f) Activate the R/O (Read-Only) attribute if it is not on (attributes are toggled with the Spacebar). Do the same thing for the Sys (System) and Hid (Hidden) attributes.
 - g) Go to the second line and repeat this process (b through f) for the second file. The one difference is that you must calculate the starting cluster for the second file this way:

Divide the file size of the first system file by the cluster size you calculated earlier. If there is a remainder, round your result up. This is how many clusters the first file takes. Add 1 to this number, and input the result as the starting cluster for the second file.
5. Write these changes to the disk.

Step 4: Adjust the File Allocation Table (FAT).

The FAT now must be adjusted to allocate the space you have provided for the System files. This could not be done until you knew the starting cluster of the second system file because that must be updated in the FAT.

Step 5: Transfer the system files.

1. Write down the starting cluster number of the second System file (as you calculated in the last step). Determine how many clusters this file takes by the same method you used for determining how many clusters the first file took (step 3—part 4—paragraph g). Write this down.
2. Select the FAT as your object and switch to the FAT view.
3. Starting at Cluster 2, create a chain for the first system file. This is done by putting a 3 on the Cluster 2 field, a 4 on the Cluster 3 field, and so on. Count the total number of clusters in your chain (don't forget to count the first one—cluster 2), until you have the same number as you calculated should be in the file. Terminate the chain by pressing e on the last cluster's field. This will result in <EOF> appearing on that spot.
4. The next cluster should be the same one you chose as the starting cluster for the second file. Repeat the process in Step 4—part 3 for this file, stopping when you have reached its full size, and terminate it with <EOF>.
5. Write these changes to the disk.

Although we have set up the FAT and the root directory for the system files, we have not yet put them on the disk. Usually this would be done with the SYS C: command, which would re-write the Boot Program as well as the system files. Try that first. If it does not work, use the following steps to move the files to the hard disk.

1. Start DiskEdit. Put a bootable floppy in Drive A:, and select that as your object.
2. Split the window, and select the root directory of Drive C: as your second object.
3. Look in the FAT of Drive A: to determine which clusters belong to the first system file. Choose those clusters as your object for the first window.

PARTITION TABLE EDITING

4. Write the clusters to Drive C:. (Use Alt-T for the Tools menu, WRITE TO... Cluster Mode, Drive C:.) You will be prompted for the starting cluster. For the first system file, the starting cluster is 2, which should be indicated in the directory display in your second window (the C: window).
5. Go back to the FAT of Drive A: and determine which clusters belong to the second system file. Choose that range of clusters as your object.
6. Write these clusters to C: also, this time using the Starting Cluster for the second file in the C: window (which you calculated earlier) as the starting cluster to write from.

The system files are now on the hard disk. If the system still does not boot, check the partition table to verify that the Boot field is set to YES for this disk.

The partition table is located at the very beginning of the physical disk. It lets the computer know that one or more logical drives are installed on that disk and that one may be bootable (this must be the first partition unless you are using an operating system such as PC-MOS.) The partition table will tell the computer where the Boot Record of a logical disk and where any Extended partitions are located. If the partition table is missing or has inaccurate information about the location of a drive's Boot Record, DOS will report "Invalid Drive Specification" when you try to access that particular drive.

If you suspect that some information may be corrupt or inaccurate in your Partition Table, use the following procedure.

1. Run DiskEdit and select the physical disk whose partition table you wish to edit as your object. From the Object menu again, choose the Partition Table. This will give you the partition table for that disk. You may need to switch to the Partition Table view.
2. We recommend saving a copy of the partition table to another disk. Use the WRITE TO... option from the Tools menu to do this.



NOTE: Save this to a floppy disk. If you merely save it to another partition of the hard disk, or, depending on the partitioning scheme, to another hard disk, you may not be able to access it when you need it. Write it in file mode.

If you need to restore the partition table from this copy, select the file you created as your object and, using the **WRITE TO...** option, write it to the hard disk in question, in Absolute Sector mode, with the default numbers.

System

DiskEdit can recalculate the partition coordinates (using the **RECALCULATE PARTITION** option on the Tools menu) as long as the calculable columns have been zeroed out.

The field headings are as follows:

This reflects the type of partition installed. Use the Spacebar to toggle among choices.

DOS-12 or DOS-16 specifies a DOS 12 or 16 bit FAT. In DOS 2.xx all disks have a 12-bit FAT. In DOS 3.xx and 4.xx most disks under 20MB have a 12-bit FAT. All disks 20-32MB and above have a 16-bit FAT.

BIGDOS specifies a DOS partition greater than 32MB, under DOS 4.0 and higher, Compaq DOS 3.31, and similar DOS versions.

EXTEND specifies a DOS extended partition table. This is where you would find the table for the next partition in a normal DOS disk.

DM specifies a Disk Manager partition.

Speed specifies a SpeedStor partition.

Boot: This is YES or NO, depending on whether or not this is a bootable partition. Only one partition can be bootable.

Starting Location

This tells you where the partition begins, and is where the actual DOS boot record is located. If you don't know where the partition begins, you can search the disk for a string of text that is unique to the boot record of your version of DOS such as "no system", "non-system disk", "dos x.x" where x.x is your version of DOS, or by searching for the FATs (by searching for "F8 FF FF" in hexadecimal) and choosing the previous physical sector. The boot record may not be in the previous sector in some versions of DOS or third party partitioning software.

Ending Location

This is the end of the partition. Partitions are not allowed to overlap, so the end of one partition must come before the beginning of the next.

Relative Sectors

This is the number of unused sectors between the physical beginning of the disk (or partition on extended partitions) and the Starting Location where the Boot Record is located.

Number of Sectors

This is the total number of logical sectors on the partition. To figure this out use the following formula:

$(\text{Ending Cylinder} - \text{Starting Cylinder} + 1) * \text{Sides} * \text{Sectors from the legal values you wrote down above.}$



NOTE: Changes will not take effect until the computer has been rebooted.

FIXING A BAD TRACK 0

This procedure will only work on disks with 17 sectors per track. It is not recommended for use on SCSI or ESDI drives. Some RLL controllers write special information onto the first track of the disk; this information is needed for the controller to operate properly. Check with the manufacturer of your disk drive if you think this will apply to your drive. If you are not sure how many sectors per track your disk has, run SysInfo, choose the Disk Characteristics menu, and select the drive in question. The number you are after is in the Physical Characteristics box, called Sectors per Track. Or, you can run Calibrate and that will tell you about your drive.

Step 1: Get a good Master Boot Program from another computer.

If you don't have another computer for this purpose available, we will use a different method to fix the Master Boot Program later.

1. Go to a computer that has the same version of DOS installed on it as does your damaged computer.
2. Turn on or reboot the computer to the C: drive.
3. Create a bootable floppy disk (FORMAT A: /S).
4. Start DiskEdit on the working computer, select PHYSICAL SECTOR from the Object menu, and enter 1 in the last text box.
5. Select WRITE TO... from the Tools menu, choose "to a File..." and type A:MASTER for the filename.
6. Exit to DOS.



NOTE: The first three steps apply to both AT-class and XT-class computers (XT-class computers have 8088, 8086, or V-20 chips in them). After completing the first three steps, skip to either Method 1 or Method 2 depending on your computer type. If you are not certain what kind of computer you have, check the processor reported by SysInfo.

Step 2: Create TRACK-0.COM.

1. On a working system, start the DOS program DEBUG. This is on your DOS disk or in your DOS directory.
2. When you get the '-' prompt, put the disk on which you created MASTER in Step 1 into Drive A:.
3. Use the information below to fill in your program, adding only the data that appears in boldface. The x's that are shown below are not going to appear as x's, but as numbers that are different for each system DEBUG is run on. If you make a mistake along the way, simply press Enter, type Q, press Enter, and start again by typing DEBUG and pressing Enter.

Method 1: AT-Class Computers

For an AT-type computer (80286, 80386, 80486 or equivalent), use the following code:

```
-a <ENTER>

xxxx:0100  MOV AX,0511  <ENTER>
xxxx:0103  MOV CX,01   <ENTER>
xxxx:0106  MOV DX,80   <ENTER>

        (This is the drive #. The default is C:.
        Change it if you have a different drive.
        80 = C:, 81 = D:, 82 = E:, 83 = F:, etc.)

xxxx:0109  MOV BX,200  <ENTER>
xxxx:010C  INT 13      <ENTER>
xxxx:010E  INT 20      <ENTER>
xxxx:0110  <ENTER>
-E200      <ENTER>

        (Press the <SpaceBar> after each number)

xxxx:0200  xx.00 xx.01 xx.00 xx.07 xx.00 xx.0D xx.00 xx.02
xxxx:0208  xx.00 xx.08 xx.00 xx.0E xx.00 xx.03 xx.00 xx.09
xxxx:0210  xx.00 xx.0F xx.00 xx.04 xx.00 xx.0A xx.00 xx.10
xxxx:0218  xx.00 xx.05 xx.00 xx.0B xx.00 xx.11 xx.00 xx.06
xxxx:0220  xx.00 xx.0C      <ENTER>

-N TRACK-0.COM  <ENTER>

-RCX  <ENTER>

CX 0000

:222  <ENTER>

-W  <ENTER>

Writing 0222 bytes

-Q  <ENTER>
```


Method 2: XT-Class Computers

For an XT-type computer (8086/8088 or equivalent), use the following code:

```
-A <ENTER>
xxxx:0100  MOV AX,0505  <ENTER>
xxxx:0103  MOV CX,01    <ENTER>
xxxx:0106  MOV DX,80    <ENTER>

                (This is the drive #. The default is C:.
                Change it if you have a different drive.
                80 = C:, 81 = D:, 82 = E:, 83 = F:, etc.)

xxxx:0109  INT 13       <ENTER>
xxxx:010C  INT 20       <ENTER>
xxxx:010E  <ENTER>

-N TRACK-0.COM  <ENTER>

-RCX          <ENTER>

CX 0000

:10E         <ENTER>

-W           <ENTER>

Writing 010E bytes

-Q           <ENTER>
```



STOP

WARNING: This is a dangerous program. Do not run it on a computer with a good hard disk. Delete or rename the file TRACK-0.COM when you are finished using it to prevent it from being accidentally run on a good disk.

Step 3: Format track 0 on your troubled computer.

1. Boot your troubled computer with the MASTER disk you created earlier.
2. Run the program TRACK-0.COM on your troubled computer: type TRACK-0 and press Enter at the A: prompt.

This program has just done a low-level format on track 0 of your damaged hard disk.

Step 4: Copy the good Master Boot Program to the damaged disk.

1. Start DiskEdit (from diskette), put the disk with MASTER back into the A: drive, and select the file MASTER as your object.
2. Choose the WRITE TO... option (from the Tools menu). Select "To Physical sectors", choose Hard Drive 1, and accept the defaults. Write the item.
3. Exit to DOS.

Step 5: Finishing touches

Remove your floppy disk from the A: drive and reboot the computer. If you can boot up off the hard disk, then this procedure was successful and you are finished.

If the system did not boot up off the hard disk, reboot with a DOS system disk in the A: drive. Try to log onto the hard drive.

If you get the C: prompt, then you were successful with fixing track 0, but your DOS system files on your hard disk are either corrupted or missing. Try fixing these first with the Make a Disk Bootable program (run Disk Tools), and, if that fails, by using the procedure Fixing the DOS System Files earlier in this Troubleshooter chapter.

If you get an error message that says "Invalid drive specification", then either the partition table needs to be edited or you have a physically damaged disk. Try running the Norton Disk Doctor. If NDD does not find or fix the problem, try the Partition Table Editing procedure in this Troubleshooter chapter.

LOW-LEVEL FORMAT FOR AN XT COMPUTER

1. Type A:DEBUG from your DOS disk.
2. Type g=c800:5 from the '-' prompt to call up the BIOS format.
3. Select the interleave that will give your disk the fastest access time. The standard for an XT would be a 3:1 interleave, although you might consult with the manufacturer of your hardware to verify this.

GENERAL DISK INFORMATION

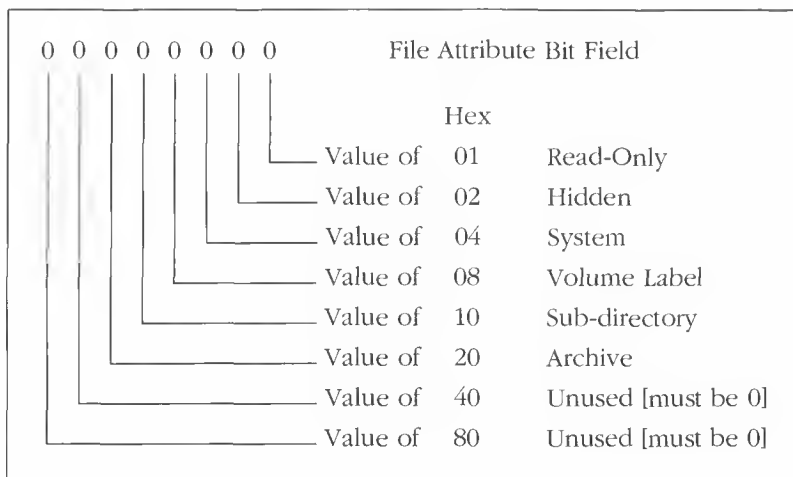
Disk Size	Disk Type	Sectrs perTrk	Media Dscrpt	Cluster Size	Sector location ranges (DOS 3.xx)				
					Boot	FAT 1	FAT 2	Root	Data Area
160K	5 ¹ / ₄ " SS	8	FE	512	0	1	2	3-6	7-319
180K	5 ¹ / ₄ " SD	9	FC	512	0	1-2	3-4	5-8	9-359
320K	5 ¹ / ₄ " DS	8	FF	1024	0	1	2	3-9	10-639
360K	5 ¹ / ₄ " DD	9	FD	1024	0	1-2	3-4	5-11	12-719
720K	3 ¹ / ₂ " DD	9	F9	1024	0	1-3	4-6	7-13	14-1439
1.2M	5 ¹ / ₄ " DQ	15	F9	512	0	1-7	8-14	15-28	29-2399
1.44M	3 ¹ / ₂ " DQ	18	F0	512	0	1-9	10-18	19-32	33-2879
10M	Hard	17	F8	4096	0	1-8	9-16	17-48	49-20720
20M	Hard	17	F8	2048 *	0	1-40	41-80	81-112	113-40457
21M	Hard	17	F8	2048 *	0	1-41	42-82	83-114	115-41666
30M	Hard	17	F8	2048 *	0	1-60	61-120	121-152	153-60157
31M	Hard	17	F8	2048 *	0	1-61	62-122	123-154	155-62202
32M	Hard	17	F8	2048 *	0	1-63	64-126	127-158	159-63856
33M	Hard	17	F8	2048 *	0	1-64	65-128	129-160	161-65516
60M	Hard	17	F8	2048 *	0	1-118	119-236	237-268	269-120596

* Cluster size is 8192 if disk is FDISKed/FORMATTED under DOS 2.xx

Absolute Sector starting location by: Side, Cylinder, Sector					
Disk	Boot record	FAT 1	FAT 2	Root directory	Data area
160K	0,0,1	0,0,2	0,0,3	0,0,4	0,0,8
180K	0,0,1	0,0,2	0,0,4	0,0,6	0,1,1
320K	0,0,1	0,0,2	0,0,3	0,0,4	1,0,3
360K	0,0,1	0,0,2	0,0,4	0,0,6	1,0,4
720K	0,0,1	0,0,2	0,0,5	0,0,8	1,0,6
1.2M	0,0,1	0,0,2	0,0,9	1,0,1	1,0,15
1.44M	0,0,1	0,0,2	0,0,11	1,0,2	0,1,1

FILE ATTRIBUTES

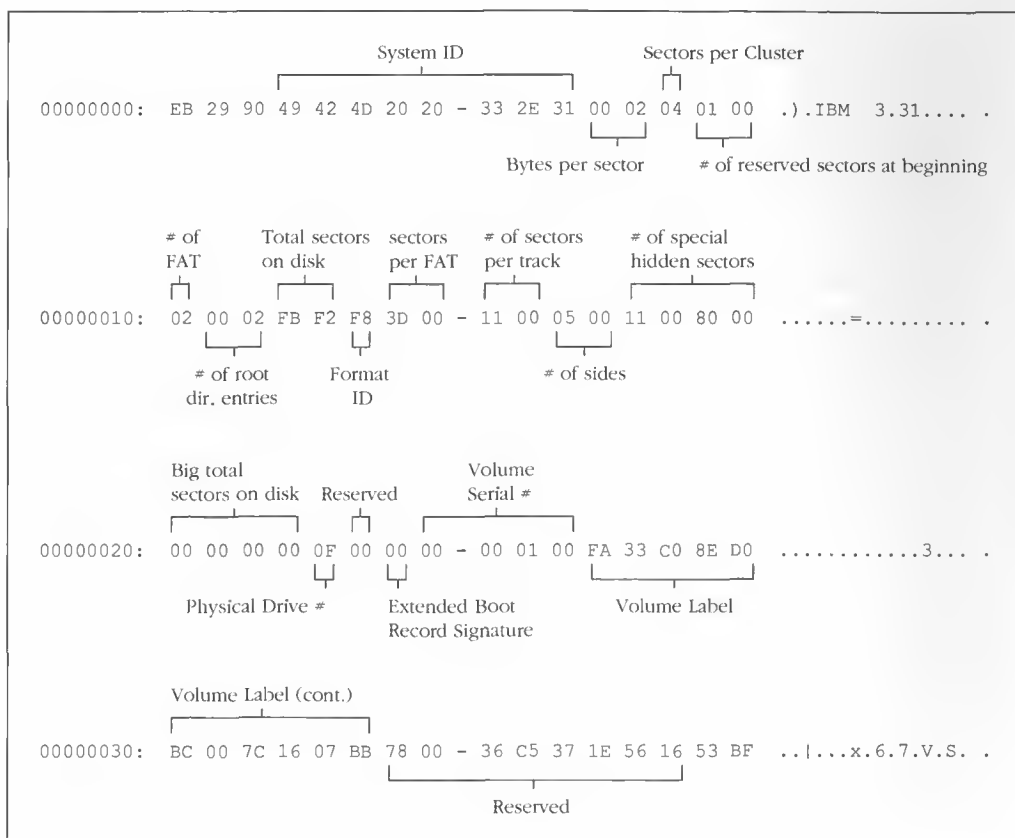
The File Attribute is the 12th byte of the directory field, which is 32 bytes long. A byte is made up of 8 bits. Each bit of the attribute byte has a different meaning, which is outlined below (for a local hard disk).



BOOT RECORD

See also the Norton Disk Doctor's "Diagnose Disk" feature and the procedure on Boot Record Editing.

Here is an example of a Boot Record for a 31MB hard disk, IBM disk type 8.



DISK TYPES FOR IBMS

Disk Type	# of Cylinders	# of Heads	# of Write Pre-compensation	Track to Park Heads	Sectors per Trk	Mega-bytes
1	306	4	128	305	17	10
2	615	4	300	615	17	21
3	615	6	300	615	17	31
4	940	8	512	940	17	64
5	940	6	512	940	17	48
6	615	4	FFFF	615	17	21
7	462	8	256	511	17	31
8	733	5	FFFF	733	17	31
9	900	15	FFFF	901	17	115
10	820	3	FFFF	820	17	21
11	855	5	FFFF	855	17	36
12	855	7	FFFF	855	17	51
13	306	8	128	319	17	21
14	733	7	FFFF	733	17	44
15	0	0	0	0	0	0
16	612	4	0 (All)	663	17	21
17	977	5	300	977	17	41
18	977	7	FFFF	977	17	58
19	1,024	7	512	1,023	17	61
20	733	5	300	732	17	31
21	733	7	300	732	17	44
22	733	5	300	733	17	31
23	306	4	0 (All)	336	17	10
24	612	4	305	663	17	21
25	306	4	FFFF	340	17	10
26	612	4	FFFF	670	17	21
27	698	7	300	732	17	41
28	976	5	488	977	17	41
29	306	4	0 (All)	340	17	10
30	611	4	306	663	17	21
31	732	7	300	732	17	43
32	1,023	5	FFFF	1,023	17	43

DISK TYPES FOR COMPAQS

Disk Type	# of Cylinders	# of Heads	# of Write Pre-compensation	Track to Park Heads	Sectors per Trk	Mega-bytes
1	306	4	128	305	17	10
2	615	4	128	638	17	21
3	615	6	128	615	17	31
4	1,024	8	512	1,023	17	70
5	940	6	512	939	17	48
6	697	5	128	696	17	30
7	462	8	256	511	17	31
8	925	5	128	924	17	39
9	900	15	FFFF	899	17	115
10	980	5	FFFF	980	17	42
11	925	7	128	924	17	55
12	925	9	128	924	17	71
13	612	8	256	611	17	42
14	980	4	128	980	17	33
15	0	0	0	0	0	0
16	612	4	0 (All)	612	17	21
17	980	5	128	980	17	42
18	966	6	128	966	17	49
19	1,023	8	FFFF	1,023	17	69
20	733	5	256	732	17	31
21	733	7	256	732	17	44
22	768	6	FFFF	768	17	39
23	771	6	FFFF	771	17	39
24	966	14	FFFF	966	17	115
25	966	16	FFFF	966	17	131
26	1,023	14	FFFF	1,023	17	122
27	966	10	FFFF	966	17	82
28	771	3	FFFF	771	17	20
29	578	4	FFFF	578	17	20
30	615	4	128	615	25	31
31	615	8	128	615	25	61
32	966	3	FFFF	966	34	49
33	966	5	FFFF	966	34	82
34	966	7	FFFF	966	34	115
35	966	8	FFFF	966	34	131
36	966	9	FFFF	966	34	148
37	966	5	FFFF	966	34	82
38	1,023	9	FFFF	1,023	33	152
39	1,023	11	FFFF	1,023	33	185
40	1,023	13	FFFF	1,023	33	219
41	1,023	15	FFFF	1,023	33	253
42	1,023	16	FFFF	1,023	34	278
43	756	4	FFFF	756	26	39
44	756	2	FFFF	756	26	20
45	768	4	FFFF	768	26	40
46	768	2	FFFF	768	26	20
47	966	5	128	966	25	60

Index

A

Absolute sector, *see* Sector, absolute
Access time, defined, 1-9, 1-10
Address, sector, *see* Sector address
Advanced Run-Length Limited Coding (ARLL), 1-9
Archive, *see* File attributes
ASCII, table of hexadecimal and decimal equivalents, 2-44, 2-51
Attributes, *see* File attributes
Auto View, 2-7
Average latency, *see* Latency

B

Backing up,
 object before editing, 2-25, 2-30, 2-32, 2-34, 2-36
 system area, 1-31, 2-11, 2-34
Bad disk, 3-7–3-13
 See also Disk, Data, Errors
Bad sectors, mapping, 1-23
BIOS, *see* ROM-BIOS
Block, marking and copying, 2-47
Boot column, Partition Table view, 2-35
Boot disk, partitioned hard disk as, 1-32
Boot Record,
 bytes per sector, 1-17
 defined, 1-16
 disk characteristics, 1-17
 editing corrupt or inaccurate, 1-35, 3-29–3-31

 example of, 3-46
 fixing bad sectors in, 3-7–3-13
 floppy disk, 3-30
 hard disk, 3-30
 in Partition Table, 1-32, 1-33
 number of heads, 1-17
 sectors per track, 1-17
 selecting as object, 2-47
 system area, 2-11
 total sectors, 1-17
Bootable, make a disk, 3-32–3-37
Boot Record viewer, 2-21, 2-49
 editing and formatting, 2-31–2-32
 keyboard and mouse navigation in, 2-31
 making and undoing changes, 2-32
 screen layout, 2-31
Bytes,
 filling with a specified character, 2-38–2-39
 finding differences in compared objects, 2-39, 2-41
Bytes per sector, on Boot Record, 1-17

C

Calibrate (CALIBRAT), 1-9, 1-12, 2-14
 calculating optimal interleave with 1-15
Cancel, closes a dialog box without saving changes, 2-8

- “Changes have been made to sector/cluster [number]” message, 2-23, 2-46
- Character fill, 2-28, 2-48
- Character filters, 2-8
- CHKDSK, DOS’s, 1-35, 3-6, 3-19
- Clipboard,
 - copying, pasting, and undoing, 2-37–2-38, 2-47
 - limits to size of copied block, 2-37
 - selecting and editing as object, 2-38, 2-47
- Clipboard, Object menu, 2-38, 2-47
- Cluster,
 - conflicting, 3-33
 - cross-linked files on, 3-6
 - defined, 1-18
 - displaying info, 2-19
 - finding unallocated, 3-4
 - larger the cluster greater the slack, 1-21
 - lost, 1-35
 - map of, 2-50
 - numbering, 1-6, 1-21
 - occupied by object, 2-20
 - overwritten data, 3-3
 - selecting as object for edit, 2-8, 2-12, 2-46
 - size set by DOS, 1-20
 - slack space in, 1-21
 - starting/ending number, 1-26–1-27, 3-3
 - status in FAT, 1-20, 1-24
 - values in FAT (table of), 1-27
- Cluster boundaries, exceeding, 2-27
- Cluster chain (FAT). . ., Link menu, 2-15, 2-48
- Cluster chain, 1-25–1-26, 2-7

- Cluster. . ., Object menu, 2-12, 2-46
- Command line use, Disk Editor, 2-52
- Compare Windows. . ., Tools menu, 2-41, 2-50
- Configuration settings, Disk Editor, 2-51
- COPY command, DOS’s, using with /V switch, 1-24
- Copy, Edit menu, 2-37, 2-47
- Cross-linked file, *see* File, cross-linked
- Cursor key movement, *see* Keyboard navigation
- Cyclic Redundancy Check (CRC), 1-24–1-27
- Cylinder,
 - defined, 1-5
 - numbering, 1-5–1-6

D

- Data,
 - blocks, 1-8
 - finding specific, 2-42
 - loss, 1-12, 2-4, 3-3–3-17
 - making, undoing, and saving changes, 2-3, 2-23–2-25
 - overwritten, 3-3
 - recovery, 3-1, 3-3–3-17
 - writing to another object or disk, 2-25
- Data area, 1-22, 1-29, 1-31
- Data encoding schemes, 1-8
 - Advanced Run-Length Limited Coding (ARLL), 1-9
 - Run-Length Limited Coding (RLL), 1-9
 - Modified Frequency Modulation (MFM), 1-8
- Data object, *see* Object
- Data organization, operating system’s, 1-10

- Data structure, viewing, 2-21
- Date, setting, 1-27, 2-31, 2-50
- DEBUG, using to create TRACK0.COM, 3-40–3-42
- Decimal values, converting between hexadecimal and decimal, 2-44
- Default settings, Disk Editor, 2-6
- DELETE command, DOS's, problems with, 3-17
- Diagnostic cylinder, 2-14, 2-52
- DIR command, DOS's, problems with, 3-14
- Directory,
 - bad entries, 3-1, 3-18–3-21
 - damaged parent, 3-25
 - displaying info, 2-19
 - file length and starting cluster number stored in, 1-26
 - fixes to, 3-17–3-29
 - length of erased file, 1-31
 - linking to file or FAT entry, 2-48
 - locating and fixing hidden files, 3-28
 - removing stubborn, 3-28–3-29
 - scrambled or unusable, 2-4
 - Speed Search, 2-11
 - tree structure, 3-21–3-25
 - viewing and editing, 2-11, 2-29–2-31
 - See also* Root directory, Subdirectory
- Directory. . ., Link menu, 2-15, 2-48
- Directory entry,
 - file attributes, 3-21
 - file creation date, 3-21
 - file extension, 3-21
 - file size, 3-21
 - filename, 3-21
 - starting location, 3-21
- Directory pointers, adjusting, 3-26
- Directory viewer, 2-21, 2-49
 - keyboard and mouse navigation in, 2-28
 - making and undoing changes, 2-28
 - screen layout, 2-29
- Directory, main, *see* Root directory
- Directory, root, *see* Root directory
- Discard Changes. . ., Edit menu, 2-24, 2-48
- Disk,
 - cluster and sector numbering, 1-6
 - cylinder and track numbering, 1-3, 1-5–1-6
 - data area and system area, 1-16, 1-31
 - disk controller coordinates, 1-21
 - file storage on, 1-19
 - file structure, 1-26
 - make bootable, 3-32–3-37
 - non-DOS, 2-4, 2-10
 - problem solving, 1-35, 2-4, 3-3–3-17
 - reading and editing data, 1-8, 2-4, 2-46
 - sector location ranges (table of), 3-44
 - structure (figure of), 1-17
 - unaccessible to DOS, 3-7
 - verification during copying, 1-24
 - writing changes from memory, 2-25, 2-48
 - See also* Drive, Floppy disk, Hard disk
- Disk characteristics,
 - hardware, 1-3–1-15
 - software, 1-15–1-35
- Disk controller, 1-8, 1-12, 1-21
- Disk Doctor, Norton (NDD), 1-34, 3-1, 3-14
- Disk drive, *see* Disk, Drive, Hard disk, Floppy disk

Disk Editor (DISKEDIT),
 accessing DOS from, 2-44–2-45, 2-51
 clipboard, 2-37
 command line use, 2-52
 configuring, 2-6, 2-51
 default settings, 2-6
 getting help with, 2-5
 installing and when not to install, 2-5
 maintenance mode, 2-14
 menu reference, 2-46–2-51
 operational notes, 2-52
 quitting, 2-44–2-45, 2-51
 read only mode, 2-6
 running from a floppy disk, 2-5
 shortcut keys, 2-46
 TSR programs with Disk Editor, 2-44
 using another utility while in, 2-45

Disk errors, finding and fixing, 1-35

Disk heads, 1-3–1-4, 1-6, 1-11, 1-17

Disk information, table of, 3-44

Disk space,
 discovering free, 2-20
 status in FAT, 1-21

Disk types,
 for COMPAQs, 3-48
 for IBMs, 3-47

Disk usage information, 2-20

Disk, floppy, *see* Floppy disk

DOS,
 accessing from Disk Editor, 2-44–2-45, 2-51
 CHKDSK, limitations of, 1-35
 cluster size, 1-20
 COPY, using with /V switch, 1-24
 DELETE, problems with, 3-17

 determining names and location of system files, 3-33–3-34

DIR, problems with, 3-14

erasing a file, 1-30

FDISK , 1-32, 2-26

FORMAT, 1-13, 1-16, 1-23, 1-32

RECOVER, 3-14

SET VERIFY ON, 1-24

 values in current use, 2-31

Drive Info. . . , Info menu, 2-20, 2-49

Drive,
 accessing current, 2-19, 2-49
 displaying info, 2-20, 2-49
 selecting for edit, 2-46
 See also Disk, Disk heads, Hard disk, Floppy disk

Drive. . . option, 2-10

E

Edit menu,
 Copy Ctrl-C, 2-37, 2-47
 Discard Changes. . . , 2-24, 2-48
 Fill. . . , 2-38, 2-48
 Mark Ctrl-B, 2-37, 2-47
 Paste Over Ctrl-V, 2-38, 2-48
 Read Only, 2-47
 Undo Ctrl-U, 2-25, 2-47
 Write changes. . . Ctrl-W, 2-25, 2-48

Eighth bit, displaying or not, 2-7–2-8, 2-51

Enhanced Small Device Interface (ESDI), 1-11

Erased entries, in Directory view, 2-29

Erased file character, 3-18

Errors,
 CHKDSK, 3-6, 3-19

Cyclic Redundancy Check (CRC), 1-24

“File not found,” 3-14

“Invalid drive specification,” 3-43

“No room for system files,” 3-33

“Sector not found,” 1-12

“Tree not processed past this point,”
3-19

Errors, disk, *see* Disk errors

Extended character set, *see* Graphics
characters

F

F6 hex, written during formatting, 1-23

FAT (File Allocation Table),

1st copy, 1-34, 2-47

2nd copy, 1-34, 2-47

backing up with Image, 1-31, 2-34

bad sector mapping in, 1-23

bytes per cluster, 1-21

bytes per entry, 1-33–1-34

cluster chain recorded in, 1-25–1-26

cluster management in, 1-20–1-21, 1-23,
1-24

cluster values (table of), 1-27

defined, 1-16, 1-19, 1-21

detecting and correcting problems in,
1-35

entries, 1-33–1-34, 2-33

filling bytes in, 2-28

in Partition Table, 1-33

linking directory or file to, 2-48

selecting as object, 2-47

See also Directory, Subdirectory, Root
Directory, System area

FAT viewer, 2-21, 2-49

keyboard and mouse navigation in,
2-34–2-35

making and undoing changes, 2-34

screen layout, 2-33

FDISK, DOS's, 1-32, 2-36

File Allocation Table, *see* FAT

File attributes,

archive bit, 1-28

bit fields (table of), 3-45

bit-encoded in root directory, 1-28

clearing with File Find, 3-17, 3-28

hidden bit, 1-28

in directory entry, 1-28, 2-29, 3-21

read-only bit, 1-28

setting, 2-31, 2-50, 3-17

subdirectory bit, 1-28

system bit, 1-28

volume label bit, 1-28

File creation date, in directory entry, 1-28,
3-21

File extension, 1-28, 1-29, 3-21

File Find, 3-17, 3-28

File fragmentation, exploring, 1-19, 2-20

“File not found” error, 3-14

File size,

in directory, 1-26, 1-28, 3-21

limit set by DOS, 1-34

File starting location, in directory entry,
3-21

File,

cross-linked, 1-35, 2-34

data loss in, 2-4

disappeared but not erased, 3-19

displaying info, 2-19

erased or unerased, 1-30

- erasing with DOS, 1-30
 - linking to FAT or directory entry, 2-48
 - overwritten (DOS 2.xx or DOS 3.xx), 3-3-3-5
 - problems with allocation chain, 1-35
 - range of clusters in, 2-20
 - removing stubborn or undeletable, 3-17-3-18
 - selecting as object, 2-46
 - setting attributes, 2-50
 - setting date/time, 2-50
 - storage on disk, 1-19
- Filename,
- 0 in first byte of, 1-30
 - ? in first byte of unerased, 1-30
 - changing filename, 1-30
 - displaying in Disk Editor, 2-6
 - first byte in erased, 1-30
 - in directory entry, 3-21
 - in root directory, 1-27-1-29
 - omitting from status bar, 2-51
 - period (.) character in first byte of, 1-30
 - sigma character in first byte of, 1-30
 - with hidden or strange characters, 3-17
- Fill. . ., Edit menu, 2-38, 2-48
- Find Again, shortcut key, 2-43
- Find. . ., Tools menu, 2-42, 2-50
- Finding data, 2-42
- Finding search string, 2-50
- Fixes,
- bad directory entries, 3-19-3-21
 - corrupt or damaged DOS system files, 3-32-3-37
 - corrupted directory, 3-18
 - cross-linked files, 3-6-3-7
 - directory tree structure, 3-21
 - fixing bad track 0, 3-39-4-43
 - incorrect media descriptor byte, 3-31-3-32
 - overwritten files, 3-3-3-5
 - partition table editing, 3-37-3-39
 - recovering subdirectories, 3-25-3-28
 - trashed or bad root directory, 3-14-3-17
- Floppy disk, 1-3, 1-6
- Boot Record, 3-30
 - editing, 2-3
 - media descriptor byte, 3-32
 - sectors per track, 1-8
 - See also* Disk, Drive, Hard disk
- FORMAT, DOS's, 1-13, 1-16, 1-23, 1-32
- Formats, print, *see* Print formats
- Formatting,
- bad sector mapping, 1-23
 - Boot Record, 1-23
 - F6 hex written, 1-23
 - FAT, 1-23
 - floppy disk, 1-13, 1-23
 - hard disk, 1-13
 - logical, 1-13, 1-16
 - low-level, for XT computer, 3-43
 - root directory, 1-23
 - sector addressing, 1-12
- Free space, *see* Disk space
- Function keys, for accessing view editors, 2-22
- ## G
- Gap bytes, 1-12
- Graphics characters, displaying in Disk Editor, 2-7-2-8
- Grow Window, View menu, 2-41, 2-49

H

Hard disk,

- access time, 1-9, 1-10
- accessing partition, 2-11
- changing interleave, 1-15
- data encoding schemes, 1-8, 1-9
- editing Boot Record, 3-30
- Enhanced Small Device Interface (ESDI), 1-11
- fixing damaged master boot program, 3-42
- formatting bad Track 0, 3-42
- head-disk assembly (HDA), 1-7
- large, FAT entries with, 1-34
- latency, 1-9, 1-11
- media descriptor byte, 3-32
- multiple partitions, 1-31
- multiple platter numbering, 1-7
- partitioning, 1-23, 1-32
- partitions greater than 32MB, 3-31
- performance standards, 1-9–1-12
- problems, 1-6–1-7
- root directory, *see* Root directory
- sector addressing, 1-11
- sectors per track, 1-8–1-9
- seek time, 1-9–1-10
- small, FAT entries with, 1-34
- tracks per side, 1-8
- transfer rate, 1-9, 1-12
- typical construction, 1-7
- See also* Disk, Disk heads, Drive, Floppy disk, Partition Table
- Hard formatting, *see* Formatting
- Hard sectoring, 1-12
- Head-Disk Assembly (HDA), 1-7
- Help, with Disk Editor, 2-5

Hex viewer, 2-21, 2-26, 2-49

- keyboard and mouse navigation in, 2-28
- making and undoing changes, 2-28
- screen layout, 2-27

Hex, decimal and character equivalents, 2-43, 2-51

Hexadecimal display, Hex view, 2-27

Hidden, *see* File attributes

High-order bit, *see* Eighth bit

I

IBMBIO.COM, 3-34

IBMDOS.COM, 3-34

Image,

- backing up system area with, 1-31, 2-32, 2-34, 2-36
- in AUTOEXEC.BAT file, 3-11, 3-16

IMAGE.DAT, using to fix FAT, 3-11, 3-16–3-17

Info menu, 2-19

- Drive Info. . . , 2-20, 2-49
- Map of object. . . , 2-20, 2-50
- Object info. . . , 2-49

Interleave,

- defined, 1-13–1-15
- figure of, 1-14
- optimizing, 1-15

“Invalid directory entries,” in Directory view, 2-29

“Invalid drive specification” error, 3-43

J

JOINED drives, using Disk Editor with, 2-52

K

Keyboard navigation, *see* Boot Record,
Directory, FAT, Hex, or Partition
Table view
Keyboard shortcuts, 2-9, 2-22, 2-43,
2-46–2-51

L

Latency, 1-11

Link menu,

- Cluster chain (FAT). . .Ctrl-T, 2-48
- Directory. . .Ctrl-D, 2-48
- Disk Editor, 2-15
- File. . .Ctrl-F, 2-48
- Partition. . ., 2-16, 2-48
- shortcut to using, 2-16
- Window. . ., 2-41, 2-48

Linking,

- files, directory entries, and FAT entries,
2-15
- objects, 2-7, 2-15
- changing views after, 2-16

Logical format, *see* Formatting

Logical sector, 2-8, 2-13

M

Main directory, *see* Root directory

Manual UnErase, 1-31

Map of Object. . ., Info menu, 2-20, 2-50

Mark, Edit menu, 2-37, 2-47

Master boot block, in partition table, 1-32

Master boot program, 3-30, 3-40–3-42

Media descriptor byte, 3-31–3-32

Memory dump, 2-18, 2-47

Memory, viewing, searching, and copying
data from, 2-18, 2-47

Modified Frequency Modulation (MFM),
1-8

Mouse navigation, *see* Boot Record,
Directory, FAT, Hex, or Partition
Table view

Multiple partitions, hard disks with , 1-31

N

NDD, *see* Disk Doctor

Network files, displaying, 2-19

“No room for system files” error, 3-33

Non-directory objects, in Directory view,
2-29

Non-DOS disks, reading and editing, 2-10

Norton Disk Doctor (NDD), *see* Disk
Doctor

NU.INI, 2-8, 2-51

O

Object information, displaying, 2-19

Object menu, 2-9, 2-11, 2-46–2-47

- 1st copy of FAT Alt-F1, 2-11, 2-47

- 2nd copy of FAT Alt-F2, 2-12, 2-47

- Boot Record. . .Alt-B, 2-11, 2-47

- Clipboard, 2-38, 2-47

- Cluster. . .Alt-C, 2-12, 2-46

- Drive. . .Alt-D, 2-46

- File. . .Alt-F, 2-46

- Memory Dump Alt-M, 2-47

- Partition Table. . .Alt-A, 2-11, 2-47

- Physical Sector. . .Alt-P, 2-47

- Sector. . .Alt-S, 2-47

Object,
 as Boot Record, 2-49
 as Directory, 2-49
 as FAT, 2-49
 as Partition Table, 2-49
 as text, 2-49
 backing up before editing, 2-25, 2-30, 2-32, 2-34, 2-36
 cluster map, 2-20, 2-50
 comparing two objects in split windows, 2-39-2-43
 copying to disk, 2-50
 creating a new, 2-25
 defined, 2-3
 find a specified string within, 2-50
 in hexadecimal, 2-49
 info, 2-19, 2-49
 linking to related objects, 2-48
 read only mode, 2-51
 viewing and editing, 2-3, 2-7, 2-8, 2-9, 2-21, 2-47

Operating system (DOS),
 data organization, 1-10
 directories and indexes, 1-16
 logical formatting, 1-16
 master boot block, 1-32
 options, Partition Table view, 2-34-2-35
 partition owning, 1-32, 2-34
 sector numbering, 1-22
 See also DOS, System files

Operating systems (non-DOS),
 partition owning, 1-32, 2-34-2-35
 reading and editing disks, 2-4, 2-10

Optimization, 1-9, 1-15

Orphaned clusters, *see* Clusters, lost
 Overwritten files, recovering, 3-3-3-5

P

Partition, 1-23
 number of sectors in, 2-35
 owning, 1-32, 2-34-2-35
 recalculating, 2-36, 2-50
 sector coordinates, 2-35
 See also Hard disk, Operating system, DOS

Partition. . . , Link menu, 2-16, 2-48

Partition Table, 1-16, 2-11, 3-30
 Boot Record, 1-32-1-33
 editing and interpreting, 2-34, 3-37-3-39
 fixing bad, 1-35
 linking to, 2-48
 location and length of each operating system's partition, 1-32
 location of FAT, 1-33
 master boot block, 1-32
 recalculating the partition, 2-36
 selecting as object, 2-47
 setting up and changing with FDISK, 1-32, 2-36

Partition Table viewer, 2-21, 2-49
 Boot column, 2-35
 keyboard and mouse navigation in, 2-36
 making and undoing changes, 2-36
 screen layout, 2-34
 System column, 2-34-2-35

Physical formatting, *see* Formatting

Physical disk. . . option, for editing
non-DOS disks, 2-10

Physical sector,
coordinates, 2-14
displaying info, 2-20
selecting as object, 2-14, 2-47
specifying number, 2-8
See also Sector, Sector address, Cluster,
Hard disk

Physical sector. . ., Object menu, 2-14,
2-47

Print As. . ., Tools menu, 2-17, 2-50

Print formats, 2-17, 2-50

Printing, to file or printer, 2-17, 2-50

Prompt, turning off in Disk Editor, 2-8

Q

Quick Links, 2-7

Quick Move, 2-6

Quit Disk Editor, Quit menu, 2-45, 2-51

Quit menu,
Quit Disk Editor Ctrl-Q, 2-45, 2-51
Shell to DOS, 2-44, 2-45, 2-51

Quit Prompt, 2-8

R

RAM, viewing, searching, and copying
data from, 2-18, 2-47

RAM drives, using Disk editor with, 2-52

Read Only mode, 2-6
preventing changes to object, 2-51
turning off, 2-5

Read-only, *see* File attributes

Read/write capabilities, of disk controller,
1-8

Read/write head, *see* Disk head

Recalculate Partition, Tools menu, 2-36,
2-50

RECOVER, DOS's,
recovery from, 3-14
removing from hard disk, 3-14
using UnFormat instead of, 3-14

Related objects, *see* Linking

ROM-BIOS calls, by side, track, sector,
1-22

Root directory, 1-16, 1-27, 1-35
backing up with Image, 1-31
fixing bad or trashed, 3-14-3-17, 3-24,
3-27
hard disk, 1-28-1-29
See also Directory, File attributes,
Subdirectory

Root directory entry,
DOS reserve bytes, 1-27
file attributes, 1-27
filename, 1-27
pointer to FAT, 1-27
starting cluster number, 1-27
table of, 1-28
time and date fields, 1-27

Run-Length Limited Coding (RLL), 1-9

S

Safe Format, 1-13, 1-16, 1-23, 1-32

Search string, finding within object, 2-50

Sector,
bad sector mapping, 1-23
bytes per, 1-8
changing interleave, 1-15
checksum value, 1-24

- clusters, 1-18
 - Cyclic Redundancy Check (CRC), 1-24–1-27
 - data, 1-22, 1-24
 - displaying info, 2-20
 - file storage, 1-19
 - interleave, 1-13–1-15
 - marked as bad by DOS, 1-24
 - numbering, 1-6, 1-21, 1-22
 - per track on Boot Record, 1-17
 - reformatting bad, 1-35
 - selecting as object, 2-10, 2-47
 - specifying number or range in Disk Editor, 2-8
 - status in FAT, 1-19–1-20
 - system area, 1-22
 - table of location ranges, 3-44
 - total on Boot Record, 1-17
 - See also* Physical sector, Cluster, Hard disk
 - Sector. . ., Object menu, 2-13, 2-47
 - Sector, absolute,
 - level fixes, 3-1, 3-37–3-42
 - table of starting locations, 3-44
 - Sector address, 1-11–1-13
 - Sector boundaries, 2-23, 2-27
 - “Sector not found” error message, 1-12
 - Sector preamble, 1-12, 1-13
 - Sector size, 1-8, 1-34
 - Seek time, defined, 1-9–1-10
 - Select File dialog box, on Object menu, 2-10
 - SET VERIFY ON, with DOS COPY command, 1-24
 - Set Date/Time. . ., Tools menu, 2-31, 2-50
 - Shell to DOS, Quit menu, 2-44, 2-45, 2-51
 - Shortcut keys, *see* Keyboard shortcuts
 - Shrink Window, View menu, 2-41, 2-49
 - Side numbering, by disk controller, 1-21
 - Slack space, 1-21
 - Soft sectoring, 1-12
 - Speed Search, accessing a directory with, 2-11
 - Split Window. . ., View menu, 2-40, 2-49
 - Subdirectory,
 - data area, 1-29
 - fixes to, 3-21–3-25
 - locating, 3-23
 - See also* Directory, Disk, File attributes, Root directory
 - SUBSTituted drives, using Disk editor with, 2-52
 - Switch Windows, View menu, 2-40, 2-49
 - Synchronization (sync) bytes, 1-12, 1-13
 - System area, 1-22
 - backing up with Image, 1-31
 - fixes to, 3-1, 3-30–3-37
 - location of root directory, 1-29
 - selecting for edit, 2-11
 - System column options, Partition Table view, 2-34–2-35
 - System files,
 - determining names and location of, 3-33–3-34
 - fixing damaged, 3-32–3-37
- ## T
- Terminate-and-stay-resident (TSR) programs,
 - avoid running while in Disk Editor, 2-44
 - Text display, Hex view, 2-27

Text string, searching for, 2-50, 3-3

Text viewer, 2-49

Time, setting, 1-27, 2-31, 2-50

Tools menu, 2-6

 ASCII Table. . ., 2-44, 2-51

 Auto View, 2-51

 Cancel, 2-51

 Character filters, 2-51

 Compare Window. . ., 2-41, 2-50

 Configuration. . ., 2-51

 Find Again. . . Ctrl-G, 2-50

 Find. . . Ctrl-S, 2-41, 2-50

 Hex converter, 2-43, 2-51

 OK, 2-51

 Print As. . . Ctrl-P, 2-17, 2-50

 Quick Move, 2-51

 Read Only, 2-51

 Recalculate Partition, 2-36, 2-50

 Save, 2-51

 Set Attributes. . ., 2-50

 Set Date/Time. . ., 2-31, 2-50

 Write to. . . Alt-W, 2-25, 2-50

Track,

 defined, 1-4-1-6

 fixing bad Track 0, 3-39-3-43

 numbering, 1-5-1-6, 1-21

See also Disk, Disk head, Drive, Hard
 disk, Floppy disk

Track-to-track seek time, 1-9-1-10

TRACK0.COM, creating with DEBUG,
 3-40-3-42

Transfer rate, 1-11

“Tree not processed past this point” error,
 3-19

U

Undo, Edit menu, 2-25, 2-47

UnErase, manual or automatic, 1-31

UnFormat, using instead of DOS's
 RECOVER, 3-14

(Un)Split Window. . ., View menu, 2-41,
 2-49

Unused directory entries, labeled in
 Directory view, 2-29

V

Verification, during copying, 1-24

View menu, 2-21-2-22

 (Un)Split Window Shift-F5, 2-49

 As Boot Record F7, 2-49

 As Directory F4, 2-49

 As Fat F5, 2-49

 As Hex F2, 2-49

 As Partition Table F6, 2-49

 As Text F3, 2-49

 Grow Window Shift-F6, 2-41, 2-49

 keyboard shortcuts, 2-22

 Shrink Window Shift-F7, 2-41, 2-49

 Split Window. . ., 2-40

 Switch Window Shift-F8, 2-40, 2-49

 Unsplit Window. . ., 2-41

Viewer,

 Boot Record, 2-21, 2-31-2-32, 2-49

 Directory, 2-21, 2-29-2-31, 2-49

 FAT, 2-21, 2-33-2-34, 2-49

 Hex, 2-21, 2-26-2-28, 2-49

 Partition Table, 2-21, 2-34-2-36, 2-49

 selecting appropriate, 2-21, 2-49, 2-51

Volume label, *see* File attributes,

W

Wildcards, using when entering a
filename, 2-10

Window. . ., Link menu, 2-41, 2-48

Windows, 2-40, 2-49

- comparing objects in, 2-50

- linking, 2-41, 2-48

- viewing objects in, 2-39–2-43

Wordstar files, 2-7–2-8, 2-51

Write Changes. . ., Edit menu, 2-25, 2-48

Write to. . ., Tools menu, 2-25, 2-50

X

XT computer, low-level format for, 3-43



Peter Norton

SYMANTEC.TM

Symantec Corporation
10201 Torre Avenue
Cupertino, CA 95014-2132
408/253-9600